A PORTABLE THREE-DIMENSIONAL GRAPHICS SOFTWARE PACKAGE

Homer John Rood, Sr.

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A PORTABLE THREE-DIMENSIONAL GRAPHICS SOFTWARE PACKAGE

by

Homer John Rood, Sr.

September 1978

Thesis Advisor: M. L. Cotton

Approved for public release; distribution unlimited

T191030

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A PORTABLE THREE-DIMENSIONAL COMPUTER GRAPHICS SOFTWARE PACKAGE		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; September 1978 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Homer John Rood, Sr.		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE September 1978 13. NUMBER OF PAGES 270
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Three-dimensional graphics; Hidden line elimination; Hidden surface elimination.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A portable three-dimensional computer graphics software package was developed utilizing the Fortran language. The package included the capability of displaying any object as a wire-frame image, as a wire-frame image with hidden lines removed,

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601 |

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

or as a solid figure with hidden surfaces removed. This computer graphics package provides the user with the ability to rotate, scale, and translate any part of the displayable image. It was utilized to display images on four distinct display devices with only minor software alterations. Three totally different host computers supported the four display devices.

Approved for public release; distribution unlimited

A PORTABLE THREE-DIMENSIONAL COMPUTER GRAPHICS SOFTWARE PACKAGE

by

Homer J. Rood, Sr.
Lieutenant, United States Navy
B.S., United States Naval Academy, 1972

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINERRING

from the

NAVAL POSTGRADUATE SCHOOL

September 1978

Thesis
F299
c.1

ABSTRACT

A portable three-dimensional computer graphics software package was developed utilizing the Fortran language. The package included the capability of displaying any object as a wire-frame image, as a wire-frame image with hidden lines removed, or as a solid figure with hidden surfaces removed. This computer graphics package provides the user with the ability to rotate, scale, and translate any part of the displayable image. It was utilized to display images on four distinct display devices with only minor software alterations. Three totally different host computers supported the four display devices.

TABLE OF CONTENTS

I. INTRODUCTION..... 17
II. THREE-DIMENSIONAL GRAPHICS..... 19
A. DISPLAY OF AN OBJECT AND ITS MOTION..... 19
B. IMAGE REALISM..... 25
C. GRAPHICS SOFTWARE STRUCTURE..... 29
III. GRAPHICS SOFTWARE COMPONENTS..... 33
A. OBJECT COORDINATE SYSTEM -THE DATA BASE..... 33
B. IMAGE SCALING..... 34
C. TRANSLATION..... 34
D. IMAGE ROTATION..... 35
1. X-Axis Rotation..... 35
2. Y-Axis Rotation..... 35
3. Z-Axis Rotation..... 36
4. An Arbitrary Axis of Rotation..... 36
E. COORDINATE TRANSFORMATIONS..... 41
1. Object to Eye Coordinate Transformation.. 41
2. Eye to Screen Coordinate Transformation.. 48
F. DISPLAY CLIPPING..... 50
G. HIDDEN LINE REMOVAL..... 54
H. HIDDEN SURFACE REMOVAL..... 69
I. IMAGE SHADING..... 84
IV. HARDWARE AND SOFTWARE CONSIDERATIONS..... 85
A. IMAGE DISPLAYED - ALL LINES SHOWN..... 85
B. INTERACTIVE SOFTWARE AND HARDWARE..... 88
C. HIDDEN LINE REMOVAL..... 91
D. HIDDEN SURFACE REMOVAL..... 93
V. A THREE-DIMESNIONAL GRAPHICS APPLICATION..... 95
A. PLANAR SURFACE PENETRATION..... 96
B. HIDDEN SURFACES DISPLAYED..... 97
C. TORPEDO TEST AREA SIMULATION..... 100

VI. CONCLUSIONS..... 105
Appendix A: THREE-DIMENSIONAL GRAPHICS SUBROUTINES AND FLOW CHARTS..... 107
Appendix B: APPLICATIONS PROGRAM AND SUBROUTINES..... 251
FOOTNOTES..... 268
LIST OF REFERENCES..... 269
INITAL DISTRIBUTION LIST..... 270

LIST OF FIGURES

1. TWO-D ROTATION OF A POINT..... 23

2. EYE COCRDINATE SYSTEM..... 24

3. WIRE-FRAME IMAGE..... 27

4. HIDDEN LINES REMCVED..... 28

5. N BY M DOT MATRIX SCREEN REPRESENTATION..... 31

6. DISPLAY SURFACE DIMENSIONS..... 32

7. ROTATION INTO X-Z PLANE..... 39

8. ROTATICN INTO Z-AXIS..... 40

9. EYE COORDINATE SYSTEM..... 42

10. VIEWING TRANSFORMATION..... 44

11. Z_e-AXIS ROTATED TOWARDS ORIGIN..... 47

12. PROJECTION OF DISPLAY POINTS ONTC SCREEN..... 49

13. CLIPPING COORDINATES..... 51

14. DISPLAY SCREEN DIVISION AND CODING..... 52

15. POLYGONAL CLASSIFICATIONS..... 59

16. EDGES'S ANGULAR COMPUTATION..... 60

17. DISPLAY AREA DIVISION AND CODING..... 61

18. ANGLE MAGNITUDE OF FOUR..... 64

19. EDGE DIVISION FOR PROPER ANGLE COMPUTATION..... 65

20. DEPTH COMPUTATION AT FOUR CORNERS OF WINDOW..... 66

21. POLYGONAL PENETRATION..... 67

22. HIDDEN LINE REMOVAL FLOW CHART..... 68

23. SCAN LINE INTERSECTION OF A POLYGON..... 71

24. SEGMENT CLASSIFICATION..... 72

25. ENTERING AND DEPARTING SEGMENTS..... 76

26. UPDATE OF SEGMENT BLOCK STORAGE..... 77

27. SPAN DESCRIPTIVE TERMS FOR A SEGMENT..... 79

28. SEGMENT BOX DEFINITION..... 80

29. IMPLIED EDGE GENERATION..... 82

30. HIDDEN SURFACE REMOVAL FLOW CHART..... 83

31. VIEWPOINT REQUIRED TO DISPLAY THE BACK SURFACES.... 99

32. TORPEDO TEST AREA.....103

33. TORPEDO APPROXIMATION.....104

LIST OF SYMBOLS

1. Display a Wire-Frame Image Symbols

A	distance of the viewer from the display area
B	vertical dimension of the display area
EDGE	array containing the indices of both endpoints of each edge
EDGE1	array containing the indices of the first endpoint of each edge
EDGE2	array containing the indices of the second endpoint of each edge
EDGEM	the number of screen coordinate edges
EDGEN	the number of object coordinate edges
ICHK	array used to code location of two endpoints of an edge being clipped
IR	array used to define the part of the image to be transformed
PCHANG	array used to mark points which have been transformed
POINTM	the number of screen coordinate points
PCINTN	the number of object coordinate points
POLGN	the number of polygons
POLYGN	array containing indices of the edges which describe each edge
POLYHE	array containing indices of the polygons which describe each polyhedron
SHAD	array containing the shades or colors of each polygon
VCX	Xs value of the screen center
VCY	Ys value of the screen center

VSX	x resolution divided by 2
VSY	y resolution divided by 2
VX	x object coordinate value of the viewpoint
VY	y object coordinate value of the viewpoint
VZ	z object coordinate value of the viewpoint
XE	array of x object coordinate values of the vertices
XS	array of x screen coordinate values of the vertices
YE	array of y object coordinate values of the vertices
YS	array of y screen coordinate values of the vertices
ZE	array of z object coordinate values of the vertices
ZS	array of z screen coordinate values of the vertices

2. Hidden Line Removal Symbols

DELTAT	the incremental angle of each edge
EDLINK	array containing the linked list of edges for each polygon
HIDER	the index of the surrounder polygon closest to the viewer
IBOTT	the Ys value of the bottom of the display area
ISIZEX	the number of resolution elements per vertical line of the display surface
ISIZEY	the number of verticle lines of resolution of the display surface
ISTACK	array used to stack the display window dimensions
ISTPTR	pointer to last display window added to the stack
JI1	the index of the first point of a polygon's

	edge found by GETNEX
JT2	the index of the second point of a polygon's edge found by GETNEX
LEFT	the Xs value of the left edge of the display area
NEXTED	the index of the next edge for a polygon
OLDP	the index of the last polygon
P	the index of the current polygon
POLEDG	array containg the index of the first edge of each polygon
POLLNK	array containing the linked list of intersectors and surrounders
PCLPTR	index of the first polygon
PCLYA	array of x value coefficients for each polygonal plane
POLYB	array of y value coefficients for each polygonal plane
PCLYC	array of z value coefficients for each polygonal plane
POLYD	array of coefficients of the polygonal plane's constant
PCLZMN	the point of each polygon closest to the viewer
SURRND	index of the first polygon on the surrounder list
THETA	the total angle of the current polygon for this display window
WBY	Ys value of the bottom of the display area
WLX	Xs value of the left-hand edge of the display area
WRX	Xs value of the right-hand edge of the display area
WTY	Ys value of the top edge of the display area
XN	Xs value of the first vertex of the current edge
XP	Xs value of the second vertex of the current

	edge
XSS	array of Xs coordinate values of display vectors
YN	Ys value of the first vertex of the current edge
YP	Ys value of the second vertex of the current edge
YSS	array of Ys coordinate values of display vectors
ZMAX1	the maximum left-lower Zs value of surrcunder polygons for the current display window
ZMAX2	the maximum left-upper Zs value of surrcunder polygons for the current display window
ZMAX3	the maximum right-lower Zs value of surrounder polygons for the current display window
ZMAX4	the maximum right-upper Zs value of surrounder polygons for the current display window
ZMINMX	the maximun Zs value of the current pclygon for the last display window
ZMIN1	the minimum left-lower Zs value of surrcunder polygons for the current display window
ZMIN2	the minimum left-upper Zs value of surrcunder polygons for the current display window
ZMIN3	the minimum right-lower Zs value of surrounder polygons for the current display window
ZMIN4	the minimum right-upper Zs value of surrounder polygons for the current display window
ZN	Zs value of the first vertex of the current edge
ZP	Zs value of the second vertex of the current edge

3. Hidden Surfaces removal and display symbols

EXLEFT	Xs value of left edge of the segment box
BXRGHT	Xs value of right edge of the segment box
EZLEFT	Zs value of left edge of the segment box
EZMAX	maximum Zs value of segment box
EZMIN	minimum Zs value of the segment box
EZRGHT	Zs value of right edge of the segment box
CHANGE	array used to mark polygons that have entering or exiting segments
CURSEG	index of current segment
DIV	division point of simple intersection of two segments within a span
DXLEFT	slope of Xs values of the left edge determining a segment
DXRGHT	slope of Xs values of the right edge determining a segment
DZLEFT	slope of Zs values of the left edge determining a segment
DZRGHT	slope of Zs values of the right edge determining a segment
EDGLST	index of the first edge linked in ENTLST
ENTLST	linked list of edges entering on a scan line
IACTIVE	array of the indices of the active segment blocks
IBFULL	used to indicate if a segment exceeds the x limits of the segment box
IBSEG1	index of the first segment of a simple intersection
IBSEG2	index of the second segment of a simple intersection
IBXCNT	number of segments in the box
IBXTYP	type of segment box
IFRELS	index of the next free segment block of storage

IMPLFT	used to flag that a span is bounded by an implied edge
IMPLST	index of current scan line sample point due to an implied edge
IMPLS2	index of first scan line sample point due to an implied edge
ISFULL	indicates if a segments Xs values exceed the span's Xs limits
IXSLFT	array containing the indices of the segment to left of any segment - an XSORT list
IXSRGT	array containing the indices of the segment to right of any segment - an XSORT list
IY	the scan line index
IYLEFT	array used to indicate on which scan line the left edge exits the display for each segment
IYENTR	array containing indices of the first edges which will enter on scan line IY
IYRGHT	array used to indicate on which scan line the right edge exits the display for each segment
LSTSEG	index of the last segment added to the display list for scan line IY
P	array containing the indices of the polygons common to each edge
PI	index of the current polygon
POLCHG	index of first changing polygon on CHANGE list
POLGON	array containing the index of the polygon of each segment selected to be displayed
POLSEG	array containing the indices of the first active segment for each polygon
PREVIS	index of the previous segment
SAMFRE	index of first free sample point storage location
SAMFST	index of the first sample point in list SAMX
SAMLNK	array of the indices for sample points which are ordered in a linked list

SAMLST	index cf the last sample pcint added to the list
SAMPLE	index cf the current scan line divisicn or sample point
SAMX	array cf the Xs values used to divide the current scan line
SDIV	the left-most endpoint of the current segment in a span
SEG	index of the current active segment
SEG1	index of new block of segment storage
SEGACT	index cf the first segment of the list of segments which will intersect the next span
SEGCNT	number of segments to be displayed on scan line IY
SEGFST	index of first segment on the XSORT lists
SEGLO	index of last active segment examined
SEGLST	array used to link the list of a polygon's edges
SEGOUT	index cf the first segment of the list of segments which intersect this span, but not the next
SXLEFT	Xs value of the left-most point of the current segment within the span
SXRGHT	Xs value of the right-most point of the current segment within the span
SZRGHT	Zs value of the right-most point of the current segment within the span
SZLEFT	Zs value of the left-most point of the current segment within the span
XLEFT	array of Xs values of the left edge of the active segments
XLSTSM	Xs value of last sample point
XRESL	the number of horizontal elements per scan line
XSPNLF	Xs value of the left edge of the current span
XSPNRG	Xs value of the right edge cf the current

	span
XRIGHT	array of Xs values of the right edge of the active segments
ZRIGHT	array of Zs values of the right edge of the active segments
ZLEFT	array of Zs values of the left edge of the active segments

I. INTRODUCTION

Presentation of information using computer aided video graphical or pen plotting devices has become very important and extremely useful in almost every profession. Three-dimensional computer graphics is being utilized to display air combat training in the military, xray scans of the human body in medecine, and blue prints and stress characteristics for mechanical parts in industry. It has even been used to make animated movies. Since one manufacturer of graphical machines has not cornered the market, software and hardware standards have not been established. Each machine has a different screen area, smallest resolution size, and data structure. The lack of a standard graphics language becomes a source of expensive software re-writing every time a newer more capable graphics machine is purchased.

The existing video graphical devices can be divided into these four distinct categories: direct view storage tubes; vector generator cathode ray tubes (CRT's); raster scan CRT's; and plasma panels. The first three types of devices have acheived high resolution displays and are most commonly used. As a minimum, the hard copy devices have the capability of pen plotting and the more advanced machines produce shaded images using electrostatic plotting. Except for a plasma panel device, access to the above graphic machines was readily available at the Naval Postgraduate School. Although all of these devices were supported through Fortran IV, the software for one machine exhibited little discernible similarity to that for another. Each graphics device did have its own manual, but the actual

efficient utilization of an output terminal, at or near its designed capability, was typically left for experimental realization.

The experimental process, learning one machine's software idiosyncrasies, required an amount of time which was generally not reduced when learning those of a second. The high cost of software and this lengthy device learning process were the two main incentives urging the development of a graphics software package which could be used with any display device. Since Fortran IV has become a universally supported language, it was selected as the development software. Graphical presentations normally involve one of the following four types: two-dimensional (2-D) graphs; 2-D images; three-dimensional (3-D) graphs; or 3-D images. The first two types of presentations have been well documented and the theory for both has been fully developed. The theory for and usage of 3-D graphs has also received considerable attention. Programs for their display have been published in many software languages. Additionally, the Naval Station in Keyport, Washington was interested in a real time 3-D Torpedo presentation of the torpedo test area for the Range Safety Officer.

For these reasons, the scope of this research was limited to developing a three-dimensional (image) graphics package, written in Fortran, which would develop a data set that could be displayed on any selected device. A portable graphics package would reduce software costs for interfacing with a new device to a minimal effort. Usage of a 3-D graphics software throughout a large organization, like the Navy, would reduce a programmer's learning experience to a one time effort. An individual would then become a portable expert.

II. THREE-DIMENSIONAL GRAPHICS

The proper display of a 3-D object on a 2-D surface, such as a video screen or a piece of paper, by a computer required that a complete numerical description of the object's boundaries or surfaces be supplied for processing. The two generally accepted methods used to represent 3-D objects are:

1. " surface definition using mathematical equations;
2. and surface approximation by planar polygonal mosaic."[1]

Either description required that a coordinate system be constructed to provide the numerical values.

A. DISPLAY OF AN OBJECT AND ITS MOTION

A right-handed, cartesian coordinate system provided an acceptable and most generally understood system to describe 3-D objects. The unit of measurement was aribitrary with the only requirement that it was consistent. This system has been labeled the Object coordinate system. The complexity of the mathematics required to define the surfaces of an object with equations was only surpassed by the algorithms required to project the image onto a 2-D surface or alter the viewing aspect. It was much easier, both for understanding and utilization, to consider each object as a set of one or more polyhedra. Since a polyhedron consists of four or more planar sides, called polygons, the realistic approximation of non-planar

surfaces, such as a sphere, was determined by the number of polygons used. By definition, a polygon was a flat surface, a plane, bounded by three or more connected, straight lines, called edges. Two edges intersected at a single point, a vertex of the polygon.

Using the object coordinate system, a vertex was defined by its x, y, and z values. With the origin placed at the geometric center of the object being described, the location of the vertices was simplified. By indexing each vertex as it was specified, an edge was determined by the indices of the two vertices which were its endpoints. Similarly, indexing each edge allowed the description of a polygon using the indices of the edges forming its boundary. Finally, a polyhedron was specified with the indices of the polygons which formed its planar surfaces.

With this type of object definition, object motion was implemented by moving its vertices in a linear manner. Linear movement of the vertices preserved the straightness of the polygonal edges and the structural purity of each polyhedron. Translation, which was the linear displacement of an object, was defined by the subtraction of a distance, T_x, T_y, or T_z, from the respective vertex coordinate value, x, y, or z. Thus, translation produced an entirely new set of vertex values, which could be stated as:

$$x' = x - T_x$$

$$y' = y - T_y$$

$$z' = z - T_z$$

Translation coupled with rotation of a point provided the ability to describe any 3-D motion. Rotation of a point P was most easily explained in two dimensions. Thus, in

Figure 1, P was rotated through the angle theta about the origin into the point P' by the transformation:

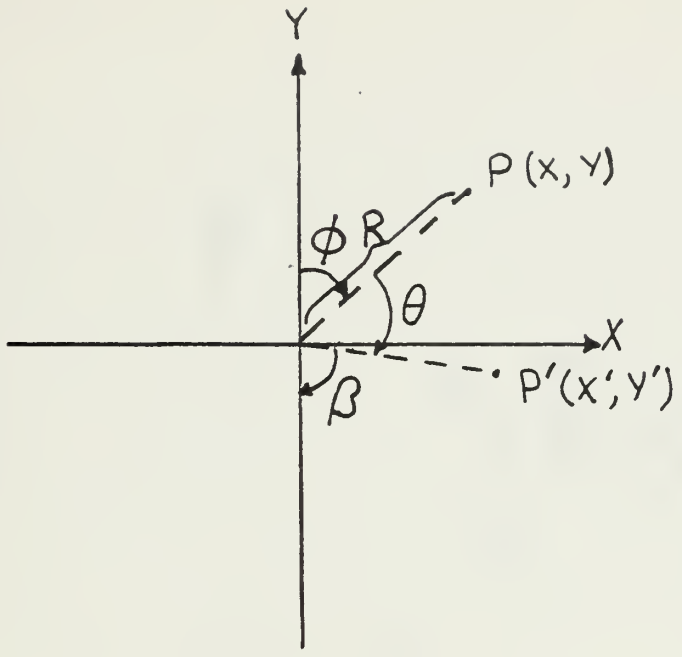
$$x' = x \cos \theta + y \sin \theta$$
$$y' = x \sin \theta - y \cos \theta$$

This derivation for 2-D rotation, as provided in Figure 1, was directly applied to 3-D rotation about the Z-axis. The angle of rotation was measured in the clockwise direction looking from the positive infinity of the axis about which a point was to be rotated. Confining the rotation of points, and thus objects, to a combination of 2-D rotations greatly simplified the computer implimentation.

The last transformation utilized to alter the viewing aspect of an object's image was scaling. This algorithm required that the vertex coordinate values be multiplied by the scale factors S_x, S_y, and S_z. Provided the scale factors were of equal magnitude, the scaling was linear and preserved the polyhedron's shape.

To project a 3-D object onto a 2-D surface required two transformations. The first, called the viewing transformation, mapped the object coordinates into a system which had its origin at the viewpoint, or the eye, of the graphic software's user. This coordinate system preserved the object's linearity and produced the image of the object as seen by the "eye". Hence, it was called the Eye coordinate system. Its Z_e-axis was used to represent, or measure, the depth of the images. The system's X_e and Y_e axes were aligned with the horizontal and vertical dimensions of the display screen, respectively. As shown in Figure 2, the viewing transformation constructed a left-handed cartesian coordinate system.

The second transformation simply projected the eye coordinate points onto the plane of the display screen. This entire transformation was easily constructed and explained in Section III. E. 2., using Figure 12.



$\theta+\phi+\beta=180°$

$\cos\beta=\cos(\pi-\phi-\theta)$ $\qquad$ $\sin\beta=\sin(\pi-\phi-\theta)$

$Y'/R=-\cos(\phi+\theta)$ $\qquad$ $X'/R=\sin(\phi+\theta)$

$=-\cos\phi\cos\theta+\sin\phi\sin\theta$ $\qquad$ $=\cos\phi\sin\theta+\sin\phi\cos\theta$

$Y'/R=-Y/R\cos\theta+X/R\sin\theta$ $\qquad$ $X'/R=Y/R\sin\theta+X/R\cos\theta$

$Y'=X\sin\theta-Y\cos\theta$ $\qquad$ $X'=X\cos\theta+Y\sin\theta$

Figure 1 - TWO-D ROTATION OF A POINT

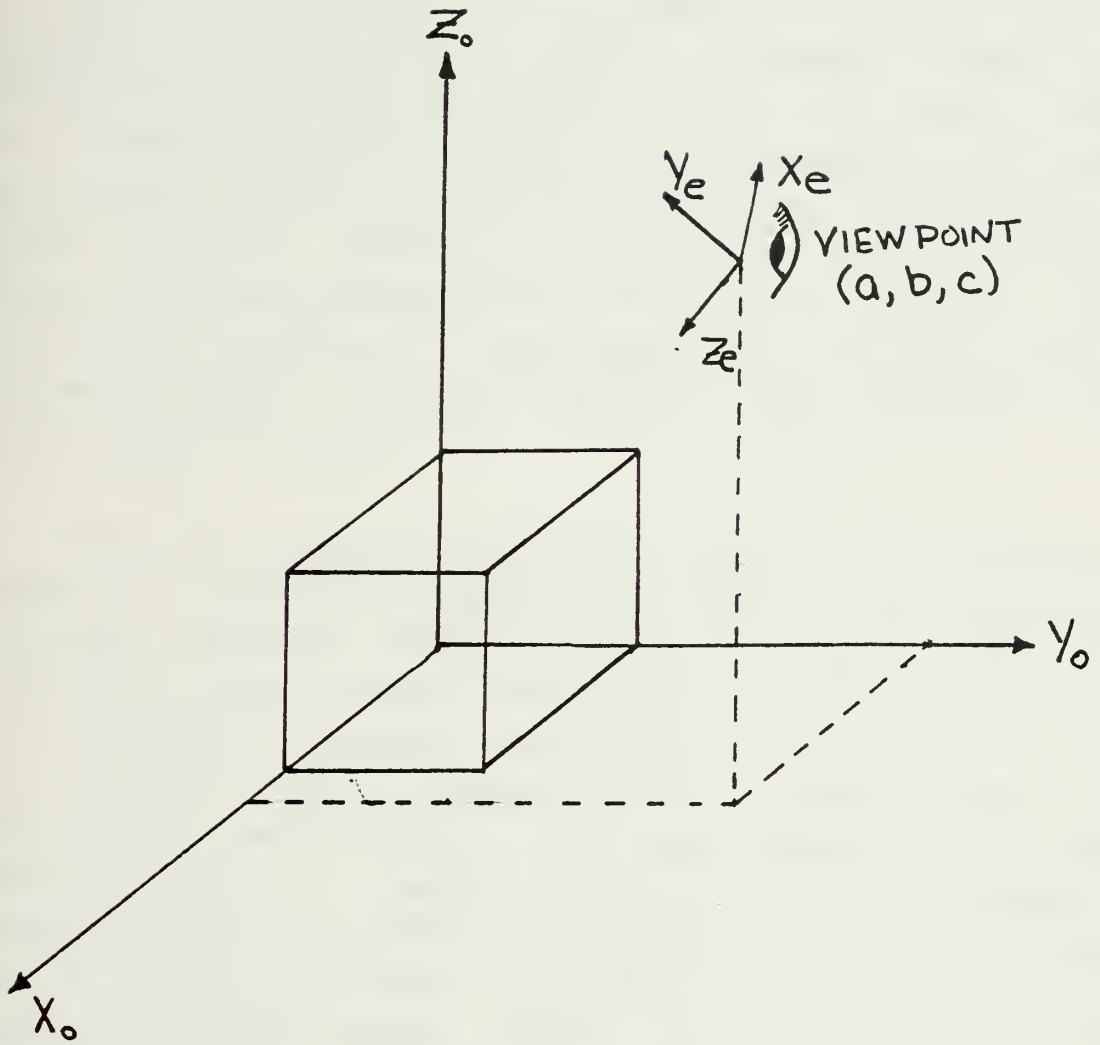


Figure 2 - EYE COORDINATE SYSTEM

E. IMAGE REALISM

The perspective projection of an object onto a screen often produced points, and thus vectors, which could not be displayed. These vectors could be located behind the observer or simply off the screen. The first case would have caused erroneous vectors to be displayed, which could not actually be seen. Typically, the second would generate program failure, because the machine could not display points, or vectors located off the screen. Therefore, the non-viewable portion of any image had to be eliminated or cut away from the viewable section. This procedure was called image clipping.

Inital 3-D displays portrayed objects as wire-framed images. A polyhedron presented in Figure 3, was shown with all edges displayed. Because this was a simple body, the dedicated observer usually recognized it as a representation of a 3-D object. However, the correct viewing aspect (i.e. which surfaces were closest to the viewer) could not positively be ascertained. For this reason, the first effort to have a computer determine a more realistic presentation was the hidden line removal algorithm. Using one of these algorithms, the display shown in Figure 4 was drawn. The proper viewing aspect was instantly apparent provided the viewer recognized the display as a 3-D image.

The next computer graphics effort to increase display realism was to have the computer generate solid polygonal surfaces. This required that the display device had a shading or full color capability. For realistic images, the computer had to possess either a software or hardware implemented hidden surface removal algorithm. Now, the

viewable image's surfaces were displayed as shaded or colored polygonal planes with user supplied shading or colors.

Most recent reasearch has concentrated on producing realistic, computer generated shading algorithms. Using the most complex shading procedures particular elements of image realism have all been acheived. However, a single universal solution to the proper shading of images has not been realized, due to different types of light sources, various material textures, and the lack of a uniform material reflectivity of light. To greatly decrease display processing time, most of the image realism programs were implemented in hardware after procedure refinement. Thus, computer graphics realism has become a function of software time available or of the cost of complex hardware.

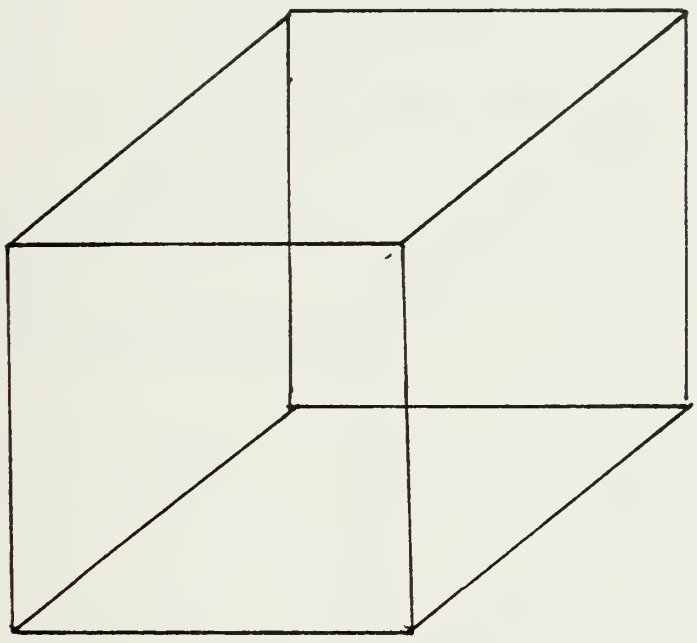


Figure 3 - WIRE-FRAME IMAGE

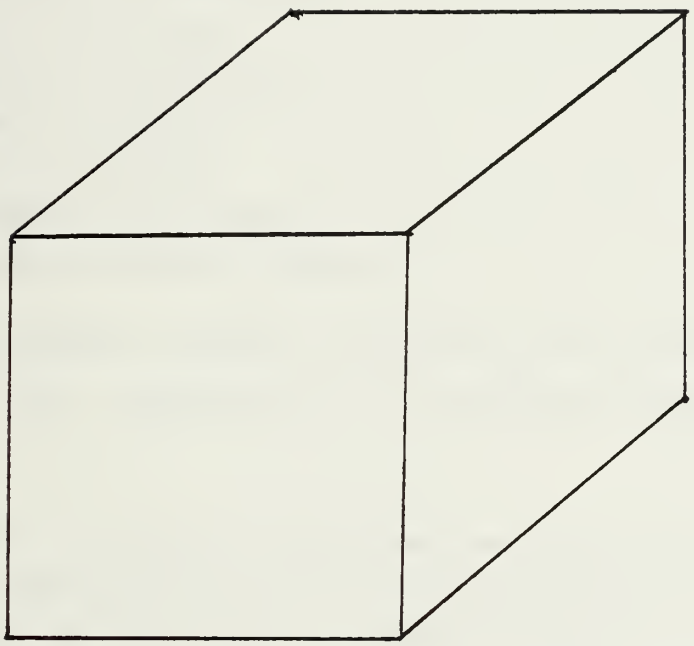


Figure 4 - HIDDEN LINES REMOVED

C. GRAPHICS SOFTWARE STRUCTURE

In order to provide a "state-of-the art", computer graphics software package, the following capabilities were determined to be the minimal requirements:

1. Object to screen coordinate transformation;
2. Clipping algorithm;
3. Image scaling;
4. Image translation;
5. Image rotation;
6. Image shading;
7. Hidden line removal;
8. Hidden surface removal.

These capabilities enabled the viewing of an image from any aspect and the generation of realistic displays in "real time".

The data structure utuilized was selected for its ease of user implementation and its appicability to the input requirements of the eight procedures above. To explain the capabilities above (in Section III.), each point, or vertex, was defined as a one by four vector, such as:

[x y z m], where m was a scale factor of the 3-D vector (normally m = 1).

This type of vector representation allowed all transformations to be defined as a four by four matrix. While the complete logic for matrix and vector utilization was provided in the Appendices of Ref. [1], a key advantage was that it facilitated the concatenation of transformations. Concatenation of matrices can be simply explained with the following example.

$[x' \; y' \; z' \; 1] = [x \; y \; z \; 1] \; \underline{A}$

$[x'' \; y'' \; z'' \; 1] = [x' \; y' \; z' \; 1] \; \underline{B}$

Can be equivalently stated as:

$[x'' \; y'' \; z'' \; 1] = [\; x \; y \; z \; 1 \;] \; \underline{T}$, where $\underline{T} = \underline{A} \; \underline{B}$

In the next section, the algorithms used to develope the graphics software package were briefly presented. Since every display device has a smallest horizontal and vertical resolution size, the proper construction of an image required the computational utilization of these machine dependent features. An easy means to visualize this concept for any device was to construct a n by m dot matrix, where n was the number of horizontal lines and m was the number of dots per line, as shown in Figure 5. Thus, to display an image required that line segments be drawn between the dots. Similarly, it was necessary to determined the screens center in order to position the image in the middle of the display surface. The following terms were used consistently to define these quanties (see Figure 6):

V_{sx} = m/2, was half of the horizontal resolution size;

V_{sy} = n/2, was half of the vertical resolution size;

V_{cx} = m/2, was the screen's horizontal center;

V_{cy} = n/2, was the screen's vertical center.

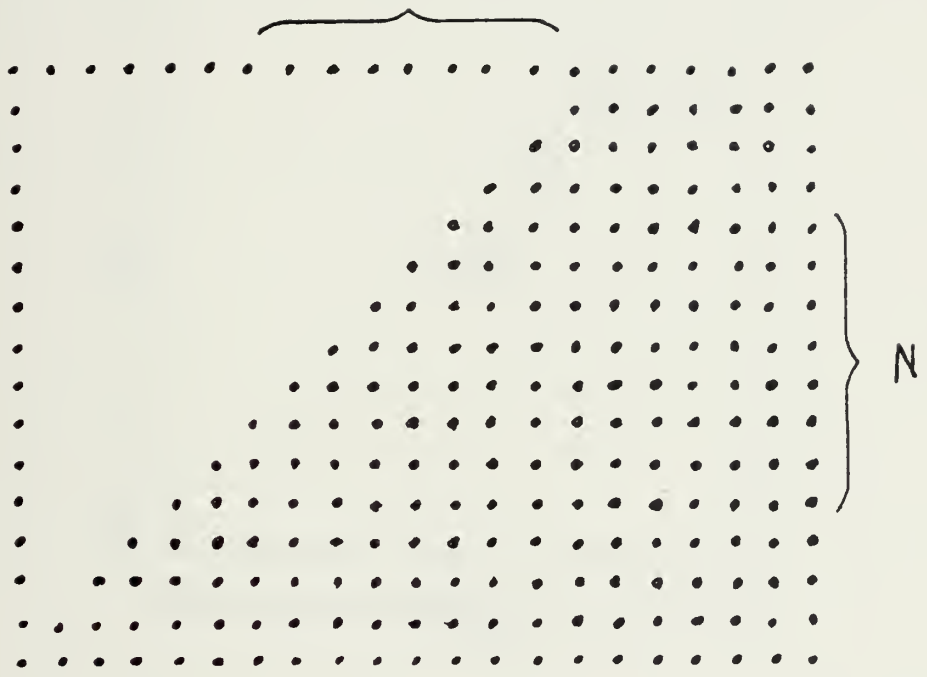


Figure 5 - N BY M DOT MATRIX SCREEN REPRESENTATION

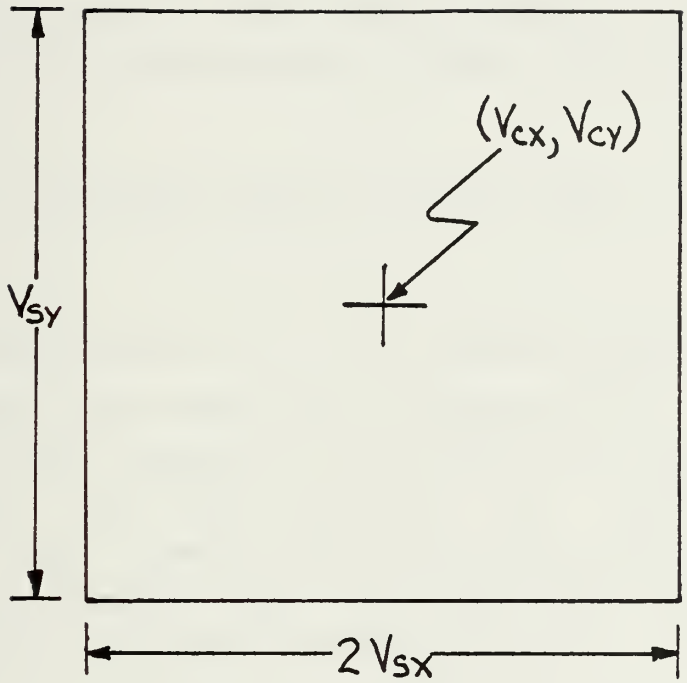


Figure 6 - DISPLAY SURFACE DIMENSIONS

III. GRAPHICS SOFTWARE COMPONENTS

In this section the algorithm for each component was briefly presented along with any figures and general flow charts which aided with the explanation. For a detailed flow chart and Fortran program listing see Appendix A.

A. OBJECT COORDINATE SYSTEM -THE DATA BASE

The description of a 3-D object with this coordinate system allowed the user total flexibility in the selection of a convenient system of measurement and origin placement. Additionally, the user selected the viewpoint, which determined the viewing axis and thus, the initial viewing aspect. The viewing axis was the line defined by the viewpoint and the object system's origin.

Image definition started with the specification of its vertices. As each vertex was input to the computer, it was assigned a consecutive index number. Similarly, as the other sets of image elements (edges, polygons, and polyhera) were input, they were consecutively indexed also. The x, y, and z object coordinate vales were stored in three real arrays, XE(i), YE(i), and ZE(i). Integer arrays were used to store indices decribing edges, polygons, and polyhedra. An edge was described by storing the index of one vextex in EDGE1(i) and the second in EDGE2(i). A polygon was defined by storing the indices of the edges which composed its boundry in the array POLYGN(i,j). The indices of these edges must be input so that:

{EDGE1(i) OR EDGE2(i)} = {EDGE1(i+1) OR EDGE2(i+1)}

The polygons which described a polyhedron were input consecutively to reduce storage requirements. Thus, a polyhedron was described by storing the index of the first and the last polygon in the array POLYHE(i,2). This data structure allowed excellent image flexibility, since any polyhedron could be rotated, scaled, or translated without altering the remaining display.

B. IMAGE SCALING

The scaling transformation multiplied the object x, y, and z values by the scale factors S_x, S_y, and S_z respectivley. If the scale factors used were not equal, image distortion resulted.

$$[X' \; Y' \; Z' \; 1] = [X \; Y \; Z \; 1]\,\underline{S}, \text{ where } \underline{S} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

C. TRANSLATION

The translation of an image, in object coordinates, represents the physical movement of a 3-D object in the x, y, or z directions by the amounts T_x, T_y, or T_z, respectively (or any combination of the three). The transformation $\underline{T}$ can be stated in matrix form as:

$$\underline{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -T_x & -T_y & -T_z & 1 \end{bmatrix}$$

D. IMAGE ROTATION

The rotation of a 3-D object was broken into four distinct categories. Each involved the rotation of the image through an angle, theta, about an axis in object coordinates. The following axes of rotation define the four categories:

1. X-Axis Rotation

The image was rotated about the X-axis through the angle theta. The angle was measured in the clockwise direction about the origin, looking towards the origin from the positive X-infinity. The transformation, $\underline{R}_x$, was defined as:

$$\underline{R}_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2. Y-Axis Rotation

The image was rotated through the angle theta about the Y-axis, where the angle was measured as stated above in

1. looking from positive Y-infinity. The transformation matrix $\underline{R}_y$, was defined as:

$$\underline{R}_y = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3. Z-Axis Rotation

The image was rotated about the Z-axis through an angle theta, which was measured as above from positive Z-infinity. The transformation matrix $\underline{R}_z$ was defined as:

$$\underline{R}_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4. An Arbitrary Axis of Rotation

The rotation of an image about an arbitrary axis required that the set of transformations below be performed on the image's object coordinates. Except for $\underline{R}_3$, these transformations were necessary to move the arbitrary axis into an axis for which a rotation algoritm (1., 2. and 3. above) already existed.

The arbitrary axis was specified by any two distinct points on it, (x,y,z) and (x',y',z'). The direction cosines for the axis, a_x, b_y, and c_z, were found by:

$a_x = (x'-x)/R$, $b_y = (y'-y)/R$, $c_z = (z'-z)/R$, where:

$$R = \sqrt{(x'-x)^2 + (y'-y)^2 + (z'-z)^2}$$

One of the points, (x,y,z), was translated to the origin using the transformation $\underline{T}$, where:

$$\underline{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x & -y & -z & 1 \end{bmatrix}$$

Now, the arbitrary axis was rotated into the Z-axis, by first rotating it about the X-axis through the angle alpha, as shown in Figure 7. This transformation, $\underline{R}_1$, which placed the arbitrary axis in the X-Z plane, was defined as:

$$\underline{R}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \text{ where:}$$

$$\cos\alpha = c_z/v$$

$$\sin\alpha = -b_y/v$$

$$v = \sqrt{b^2 + c^2}$$

The arbitrary axis was then rotated about the Y-axis through an angle beta, as shown in Figure 8, into the Z-axis. This transformation matrix, $\underline{R}_2$, was described as :

$$\underline{R}_2 = \begin{bmatrix} v & 0 & a_x & 0 \\ 0 & 1 & 0 & 0 \\ -a_x & 0 & v & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The rotation about the arbitrary axis through the angle theta was now defined as a rotation about the Z-axis. Thus, the transformation, $\underline{R}_3$, was exactly as described in 3. above for $\underline{R}_z$. The remaining task was to return the arbitrary axis, and the image, back to its original spacial position. This was accomplished by multiplying with the inverse of the transformations $\underline{T}$, $\underline{R}_1$, and $\underline{R}_2$ in the reverse order. Using the principle of concatenation, the total transformation may be stated as :

$$\underline{R}_T = \underline{T}\ \underline{R}_1\ \underline{R}_2\ \underline{R}_3\ \underline{R}_2^{-1}\ \underline{R}_1^{-1}\ \underline{T}^{-1}$$

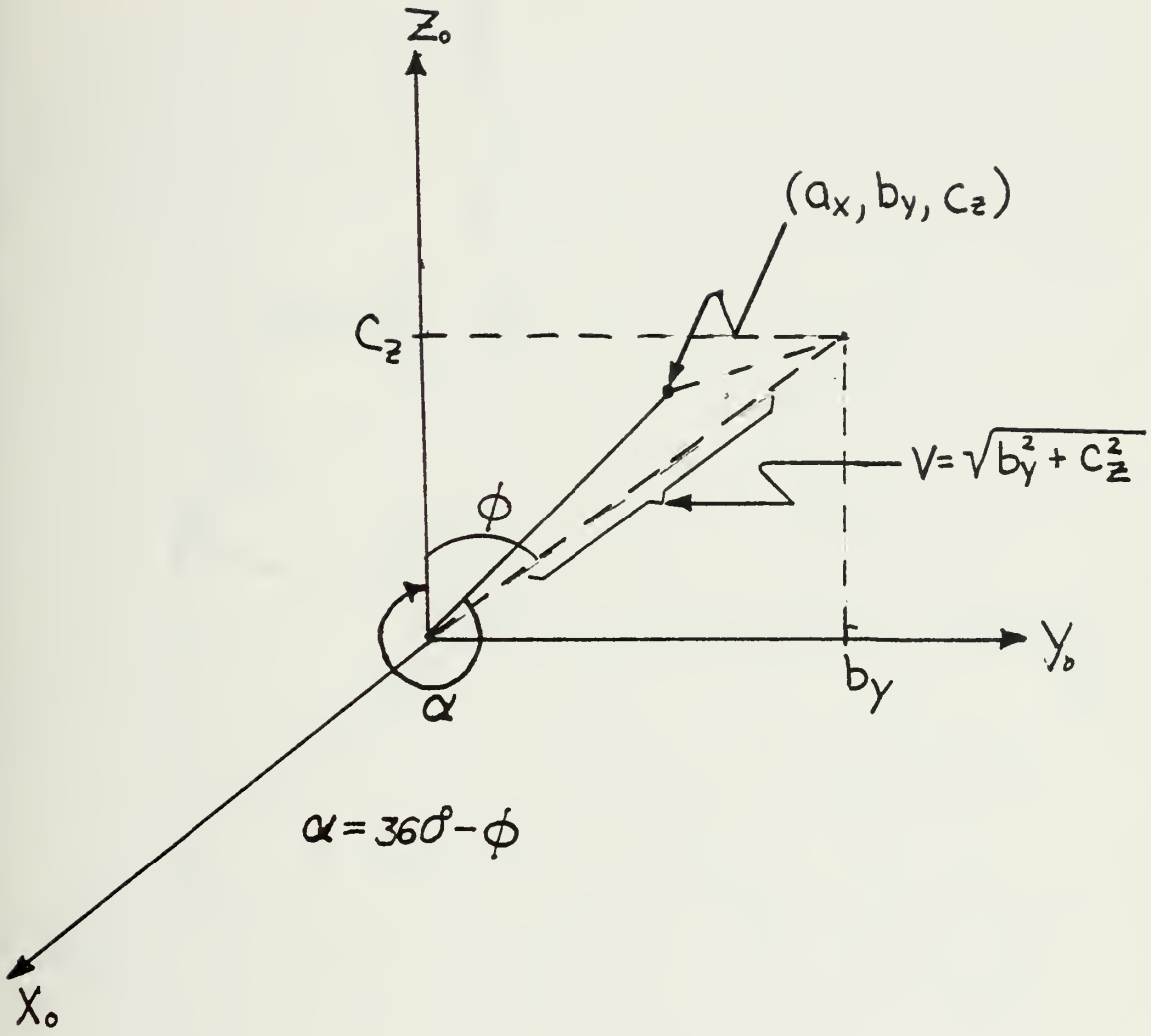


Figure 7 - ROTATION INTO X-Z PLANE

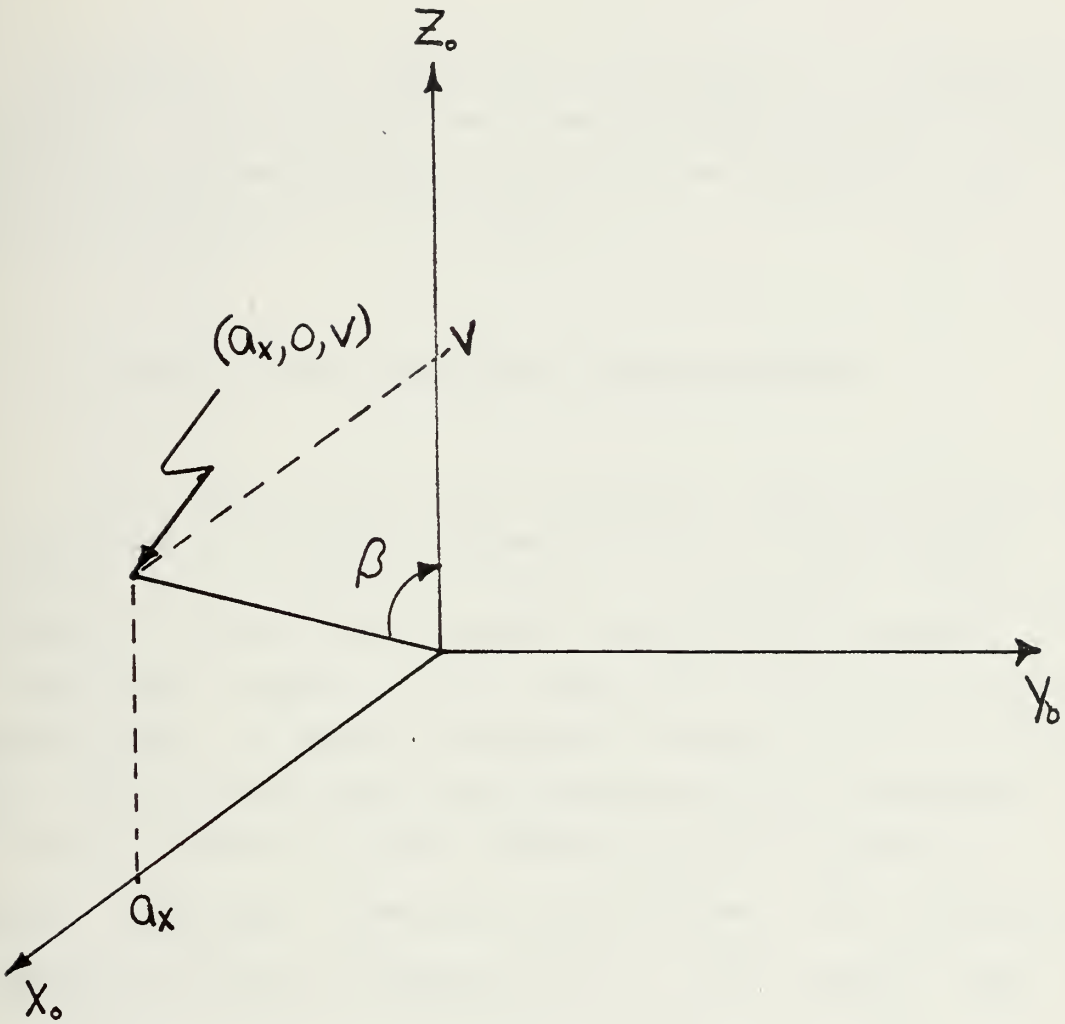


Figure 8 - ROTATION INTO Z-AXIS

E. COORDINATE TRANSFORMATIONS

To display a 3-D object on a 2-D screen, a perspective projection was performed so that an object, such as a cube, when viewed orthographically (as on the screen) gave the proper preception of a 3-D form. [3]

1. Object to Eye Coordinate Transformation

The left-handed, Eye coordinate system was utilized to determine the proper perspective view of any object. To eliminate the extremely complex viewing angle computations incurred by placing the observer close to the object (i.e. in the near field), the viewer was located an infinite distance from the object coordinate origin on the viewing axis. This allowed the rays emanating from an observers eye to all be parallel to the viewing axis, the Z_e-axis, at the object. Parallel viewing rays allowed an orthographic projection of the 3-D object onto a 2-D screen with the Z_e-axis representing viewing depth. The X_e and Y_e axes were then aligned with the screen's horizontal and vertical dimensions, respectively. The eye coordinate system, as shown in Figure 9, preserved the linearity of the image. The transformation from the object to the eye system was called the viewing transformation, $\underline{V}$, and was defined as:

$$[X_e \; Y_e \; Z_e \; 1] = [X \; Y \; Z \; 1] \; \underline{V}, \text{ where:}$$

$$\underline{V} = \underline{T}_1 \; \underline{T}_2 \; \underline{T}_3 \; \underline{T}_4$$

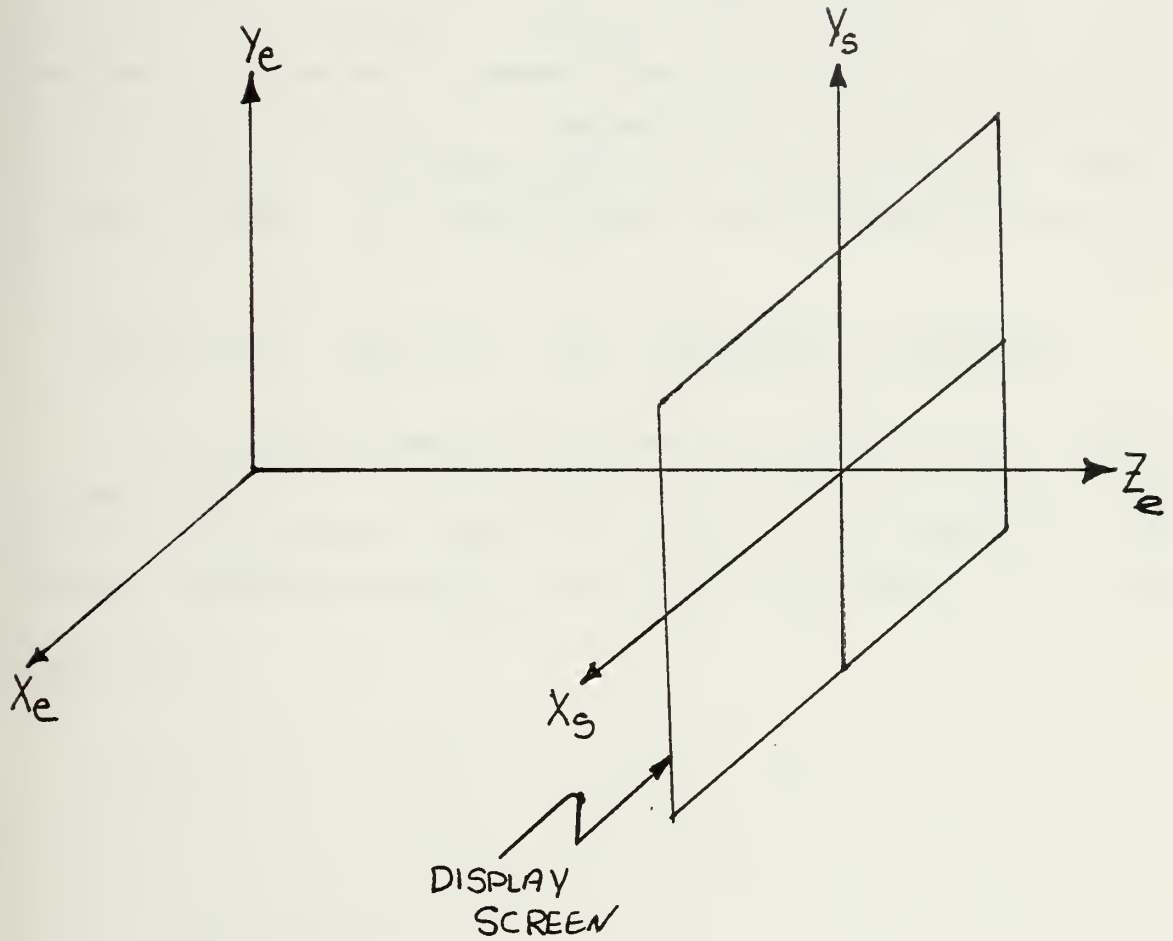


Figure 9 - EYE COORDINATE SYSTEM

Prior to this first coordinate transformation, the object coordinate values for the vertices were placed in the arrays XS(i), YS(i), and ZS(i). These arrays were used to generate the displayable information by the clipping, hidden line removal, or hidden surface removal algorithms. This prevented the original object's description from being destroyed or altered in these three procedures and allowed the graphics package to subsequently present different viewing aspects. Additionally, since clipping could remove an entire edge, the vertex indices were stored in EDGE(2,i) as:

EDGE(1,i) = EDGE1(i) and EDGE(2,i) = EDGE2(i)

The transformation matrices specifying $\underline{V}$ were formed as shown below when the viewpoint was located at (a,b,c) (in object coordinates) and the object was centered at the origin. Transformation $\underline{T}_1$ translated the viewpoint to the origin by :

$$\underline{T}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -a & -b & -c & 1 \end{bmatrix}$$

A left-handed cartesian coordinate system was formed with $\underline{T}_2$, where:

$$\underline{T}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

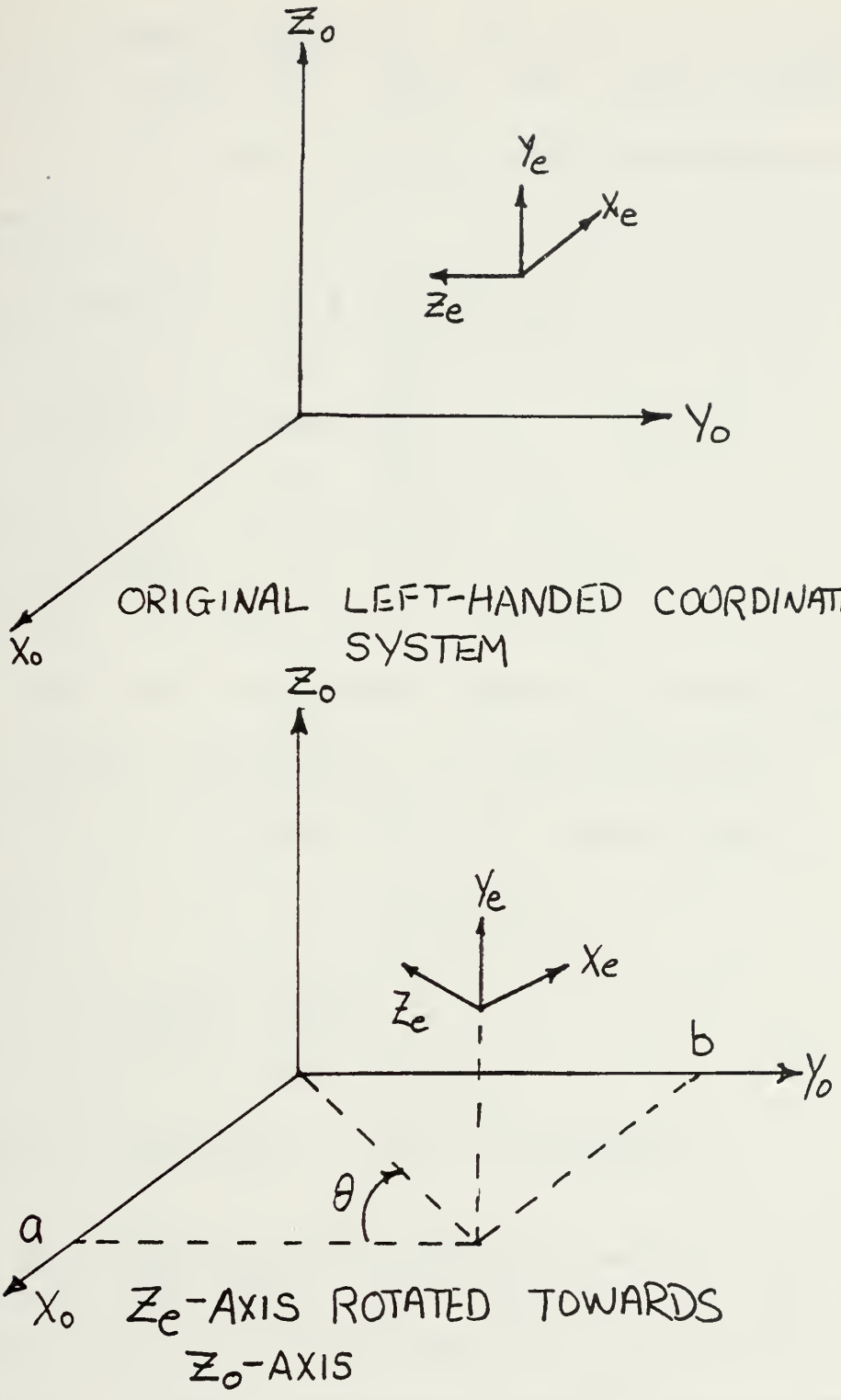


Figure 10 - VIEWING TRANSFORMATION

As shown in Figure 10, the system was rotated about the Y_e-axis through the angle theta which pointed the Z_e-axis at the point (0,0,c). The transformation $\underline{T}_3$ was specified as :

$$\underline{T}_3 = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} , \text{ where:}$$

$$\cos\theta = a/v , \quad \sin\theta = b/v$$

$$v = \sqrt{a^2 + b^2}$$

Next, the coordinate system was rotated about the X_e-axis through the angle phi, as shown in Figure 11. This pointed the Z_e-axis towards the object space origin, where:

$$\underline{T}_4 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} , \text{ for:}$$

$$\cos\phi = v/(v^2+c^2) , \quad \sin\phi = c/(v^2+c^2)$$

$_4$

This transformation arbitrarily selected the viewing axis as the line between the viewpoint and the object coordinate axis. It also placed the X_e-axis in the object system's Z = c plane. Since the object coordinate system was user defined the logical initial viewing aspect was to look at the system's origin or center. The initial view due

to the position of the X_e-axis was acceptable and basically irrelevant since the image could be rotated, translated, or scaled to provide any desired viewing aspect.

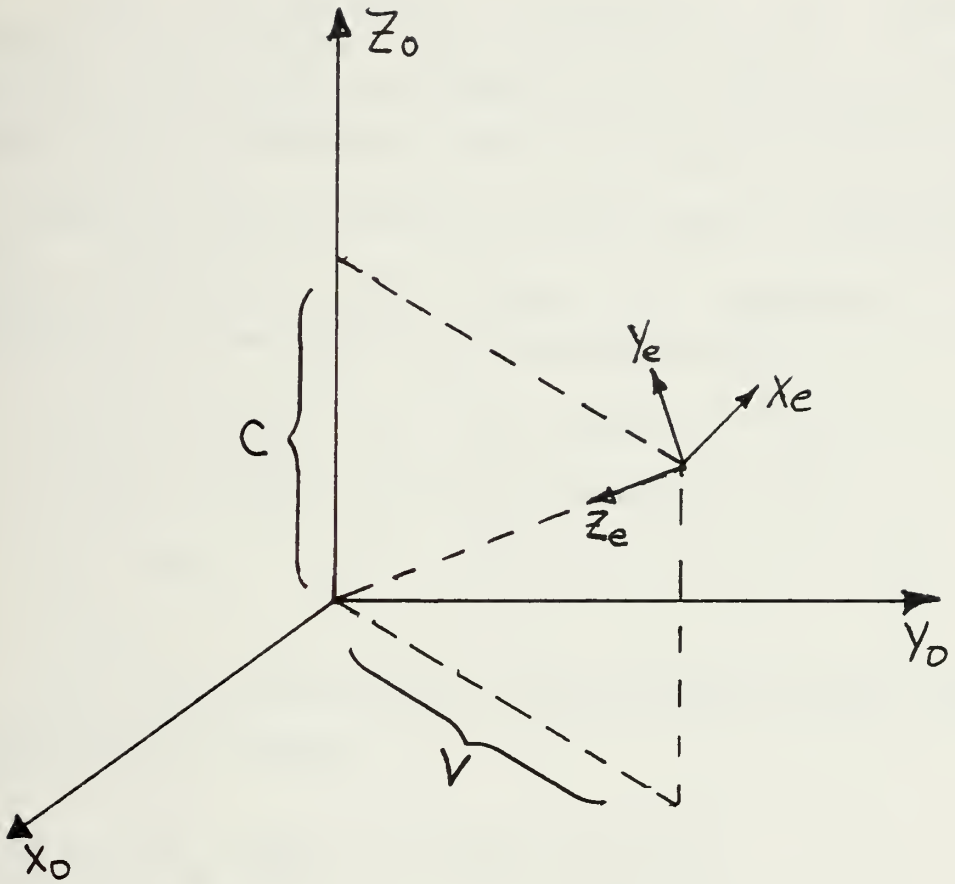


Figure 11 - Z_e-AXIS ROTATED TOWARDS ORIGIN

2. Eye to Screen Coordinate Transformation

This transformation completed the perspective projection of the 3-D object onto the screen. As shown in Figure 12, the display was generated by simply projecting an object's eye coordinates onto the plane of the screeen.[5] For a square display screen, one with equal horizontal and vertical resolution, the image was constructed without distortion, by the following transformation:

$$X_s = S_x (X_e / Z_e) V_{sx} + V_{cx}, \text{ where } S_x = a/b$$

$$Y_s = S_y (Y_e / Z_e) V_{sy} + V_{cy}, \text{ where } S_y = a/b$$

$$Z_s = -1/Z_e$$

If the viewing screen was not square then S_x and S_y were modified as shown:

$$\text{If } V_{sx} > V_{sy} \text{ then } S_x = (a/b)(V_{sy}/V_{sx})$$

$$\text{If } V_{sy} > V_{sx} \text{ then } S_y = (a/b)(V_{sx}/V_{sy})$$

This modification was necessary so that an un-distorted image could be displayed over an entire rectangular screen. Without it one display dimension, typically the horizontal, would have been elongated.

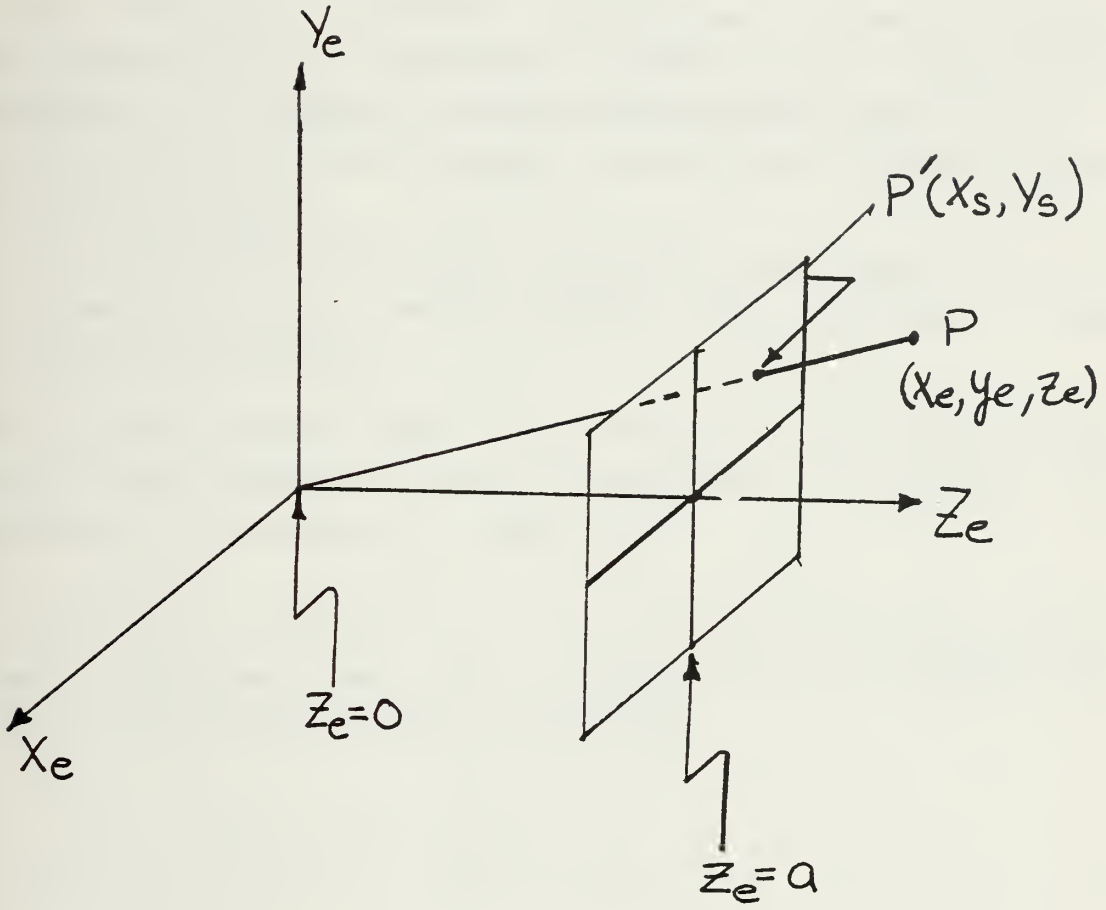


Figure 12 - PROJECTION OF DISPLAY POINTS ONTO SCREEN

F. DISPLAY CLIPPING

The clipping procedure constructed a viewing pyramid which eliminated the undesirable effects of the perspective projection from object to screen coordinates, which were:

1. points and thus objects may have been located behind the viewpoint;
2. and objects may have exceeded the limits of the viewpoint (i.e. were located off the screen - were non-displayable).[6]

The clipping of an image was performed on the image's data while it was expressed in eye coordinates to simplify the operation (as explained in Ref. [1]).

As shown in Figure 13, the geometry of the viewing pyramid dictated that for a point to be visible the following conditions must be satisfied:

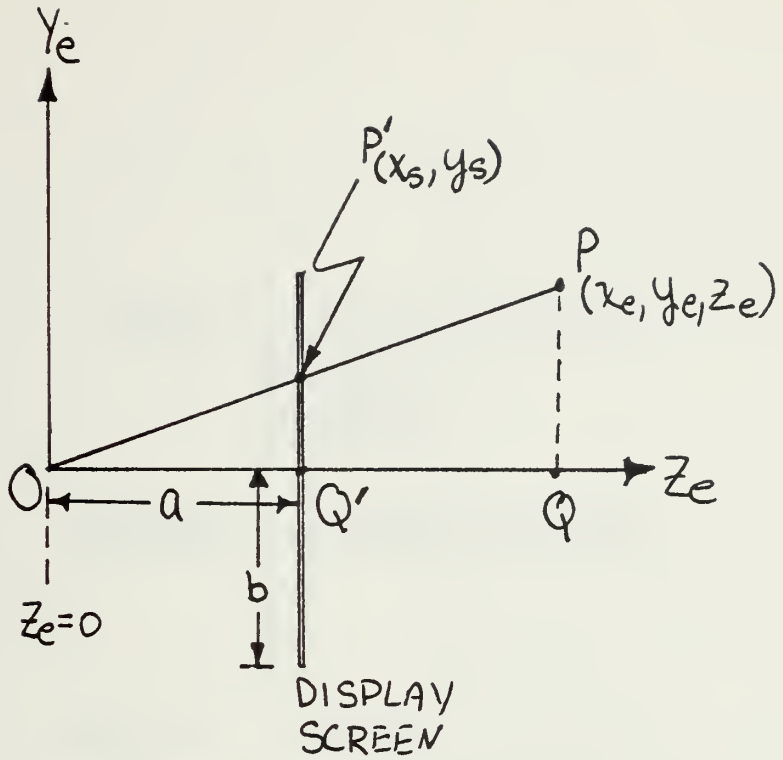
$$-Z_e \leq S_x X_e \leq Z_e \qquad (1)$$

$$-Z_e \leq S_y Y_e \leq Z_e \qquad (2)$$

Thus, a transformation from eye coordinates to a "clipping" coordinate system was described as:

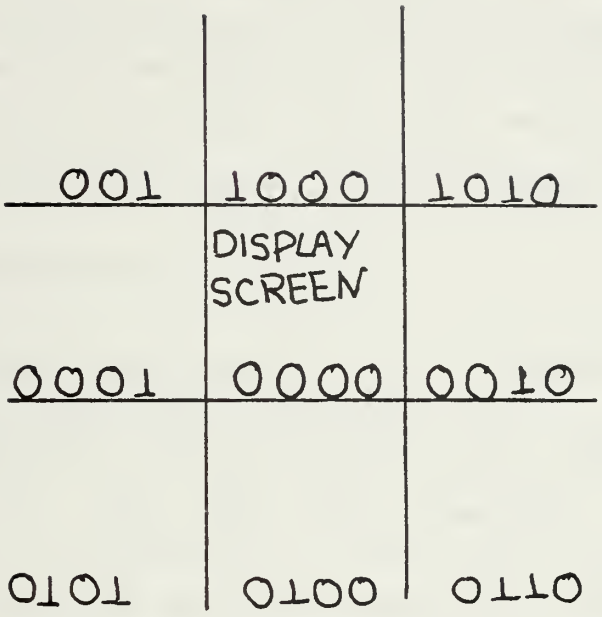
$$[X_c \; Y_c \; Z_c \; 1] = [x_e \; Y_e \; Z_e \; 1] \, \underline{N}, \quad \text{where:}$$

$$\underline{N} = \begin{bmatrix} a/b & 0 & 0 & 0 \\ 0 & a/b & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



IF $(9/b)\,|Y_e| > Z_e$ THEN POINT P' WILL BE OFF THE SCREEN

Figure 13 - CLIPPING COORDINATES



BIT 0: LEAST SIGNIFICANT BIT

BIT 0 = ICHK(I,1)

BIT 1 = ICHK(I,2)

BIT 2 = ICHK(I,3)

BIT 3 = ICHK(I,4)

BIT 3: MOST SIGNIFICANT BIT

Figure 14 - DISPLAY SCREEN DIVISICN AND CODING

This coordinate system was established to display only viewable points. Additionally, if an edge's endpoint was located outside of the viewing pyramid, this routine located a point on the edge which satisfied equations (1) and (2) above and became the new endpoint. By dividing the plane of the screen, the X_s-Y_s plane, into nine sectors (Figure 14), the location of the two verticies of an edge was determined using the inequalities (1) and (2).

Since a vertex can be used as an endpoint of several edges, the clipping procedure placed the x, y, and z values of the two endpoints into the arrays X(2), Y(2), and Z(2), respectively, as each edge was examined. Thus, if new endpoints had to be computed for this edge, the values of the original verticies were not destroyed. Since Fortran IV did not support binary operations, an integer array, ICHK(2,4), was used to code the location of each endpoint as follows:

```
If X(i) < -Z(i) then ICHK(i,1) = 1 else ICHK(i,1) = 0
If X(i) >  Z(i) then ICHK(i,1) = 1 else ICHK(i,1) = 0
If Y(i) < -Z(i) then ICHK(i,3) = 1 else ICHK(i,3) = 0
If Y(i) >  Z(i) then ICHK(i,4) = 1 else ICHK(i,4) = 0
```

If both endpoints were displayable, no action was taken and the next edge was examined. If both vertices were to the right of the viewing pyramid (or both to the left, or both above, or both below), the entire edge was deleted from the display (i.e. if ICHK(1,j) = ICHK(2,j) = 1 then discard the edge). Until both endpoints were displayable or the edge could be rejected, new points had to be computed on the edge. This computation has often been termed "pushing" the endpoint towards the display area. The pushing of the endpoint was accomplished by utilizing the 3-D, parametric

representation of a line to select that point where:

$|X_c| = |Z_c|$ if inequality (1) was not satisfied;

OR

$|Y_c| = |Z_c|$ if enequality (2) was not satisfied.

If inequality (1) was violated, the fcllowing sets of equations are used to compute the new endpcint:

If $X(1) < -Z(1)$: then:

$$t = [Z(1)+X(1)]/\{[X(1)-X(2)]-[Z(2)-Z(1)]\}$$
$$Z(1) = t * [Z(2)-Z(1)] + Z(1)$$
$$X(1) = -Z(1)$$
$$Y(1) = t * [Y(2)-Y(1)] + Y(1)$$

If $X(1) > Z(1)$ then:

$$t = [Z(1)-X(1)]/\{[X(2)-X(1)]-[Z(2)-Z(1)]\}$$
$$Z(1) = t * [Z(2)-Z(1)] + Z(1)$$
$$X(1) = Z(1)$$
$$Y(1) = t * [Y(2)-Y(1)] + Y(1)$$

When inequality (2) was violated, the equations used to compute the new point were those above with every X replaced with a Y and vice versus.

G. HIDDEN LINE REMOVAL

The method utilized to remove hidden lines from 3-D objects was developed by John Warnock at the University of Utah. The program was interpreted from a SAIL prcgram listed in Ref.[1]. This prccedure required that the cbject coordinates be transformed to eye and then to screen coordinates without any intervening clippirg of the image. The algoritm was broken into three main sections, the

Looker, the Thinker, and the Controller. The storage of the vertex indices in the array EDGE(2,j) was re-structured so that the index of a polygon's first edge could be used to link to the index of its second, and its second could link to the third edge, etc. Thus, an initializing subroutine linked each polygon's edges in the array EDLINK(i).

The concept of linked lists uses the index of edge i to produce the index of the next edge for the same polygon. Since an edge could be common to two polygon's, the first polygon to link the edge i found the storage location EDLINK(i) unused. This first polygon stored the index of its next edge, j, in EDLINK(i). Because two polygons can have at most one edge in common, the second polygon to link edge i, could not use this same storage location, EDLINK(i). Therefore, the number of edges, called edgem, was increased by one and the vertices for edge i were also stored as shown:

```
EDGE(1,EDGEM) = EDGE(1,i)
EDGE(2,EDGEM) = EDGE(2,i)
```

Additionally, the storage location EDLINK(EDGEM) was used to store the index of the next edge for this second polygon. This ordering usually doubled the storage requirements for edge definition. The vertex indicies were further ordered in the array EDGE(i,j), so that:

```
EDGE(2,j) = EDGE(1,j+1)
```

The data was structured, using linked lists (integer arays) and pointers to the first element of the list, as follows:

1. POLINK-a list of polygon indices ordered by the polygon closest to the viewer with pointer POLPTR;
2. POLEDG- contained the indices of the first edge of

each polygon. The pointer used was the polygons index;

3. EDLINK-contained the linked list of edges. The pointer to a polygons second edge was the index of its first (found in PCLEDG);

4. POLLST-a list of polygons which were determined by the Locker to be either surrounders or intersectors (which are explained below). The pointers to the list were SURRND and INTER.

An example of the usage of these linked lists was provided below:

INTER was the last polygon (index) added to the list of intersectors and thus, was at the head of the list.

POLLST(INTER) = P, where P was the index of the second polgon on the intersectors list.

POLEDG(P) = E_1, where E_1 was the index of the first edge for polygon P.

EDLINK(E_1) = E_2, which was the index of the second edge for polygon P.

EDGE(1,E_1) = V_1 and EDGE(2,E_1) = V_2, where V_1 and V_2 were the indices of the two vertices discribing edge E_1.

The concept of linked lists was utilized extensively for both the hidden line and hidden surface removal routines. In this proceedure, a display window, which was initially the entire screen, was examined against each polygon. Each window could be classified as:

1. nothing was contained in this window;

2. the information contained in this window was simple and could be displayed;

3. or the information contained in the window was too complex.

Situations 1. and 2. resulted in a successful processing of the window. The next window on the stack could then be examined. The last classification was a failure and caused the window to be divided into four windows of equal size. These new windows were then pushed onto the stack.

The first important element needed to process a display window was the computation of the depth of the plane determined by a polygon at the four corners of the window. The planar equation can be stated as:

$$Ax + By + Cz + D = 0$$

The coefficients, A, B, C, and D, can be found from the x, y, and z values of any three points contained in the plane which are not colinear. Since the corners of a window are specified as X_s and Y_s values in screen coordinates, the depth of the polygon was computed by simple substitution into the plane equation.

The last important concept needed to process a window was the classification of each polygon as:

1. an intersector of the window;
2. a surrounder of the window;
3. or disjoint from the window.

These concepts are clearly portrayed in Figure 15. The classification of all polygons was performed by the Looker.

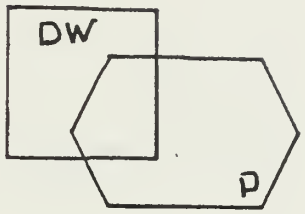
To determine whether a polygon was an intersector, it was sufficient to find any one of the polygon's edges which intersected the window. This determination was made by a clipping subroutine, which was very similar to that in F. If none of the edges intersected the window, the polygon was determined to be a surrounder or disjoint from the window by computing the angle "about the window through which each

edge passed." [6] The sum of these angles, as all edges of a polygon were processed, would equal ±360 degrees, if the polygon was a surrounder of this window, as shown in Figure 16. The actual computation of each edge's angle was implemented by dividing the X_s-Y_s plane into nine sectors, as shown in Figure 17. The window was located in the center region, and the outer eight regions were numbered as shown. The endpoints were located, just as in the clipping routine in F. , and assigned the proper sector number.

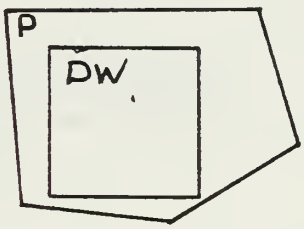
The edge's "angle" was the number of sectors which an edge entered, not counting the sector of the first endpoint. A polygon which surrounded the window had an "angle" of ± 8, and a disjoint polygon had a zero angle. Extremely complex polygons could have an angle equal to ± 16, or higher multiples of eight, by surrounding the window two or more times. However, usage of such complex polygons was unnecessary to construct any image. Because the incremental angle ( ) was defined as the difference between the sector values of the two endpoints, one problem of computing an edge's angle occurred when the magnitude was greater than four. Since no linear edge could enter more than four sectors (starting from the sector of the first endpoint), the edge's angle was adjusted when the magnitude was greater than four by :

If $\Delta\alpha > 4$ then $\Delta\alpha = \Delta\alpha - 8$
If $\Delta\alpha < -4$ then $\Delta\alpha = \Delta\alpha + 8$

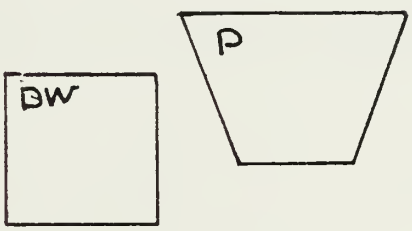
DW = DISPLAY WINDOW
P = POLYGON



INTERSECTOR

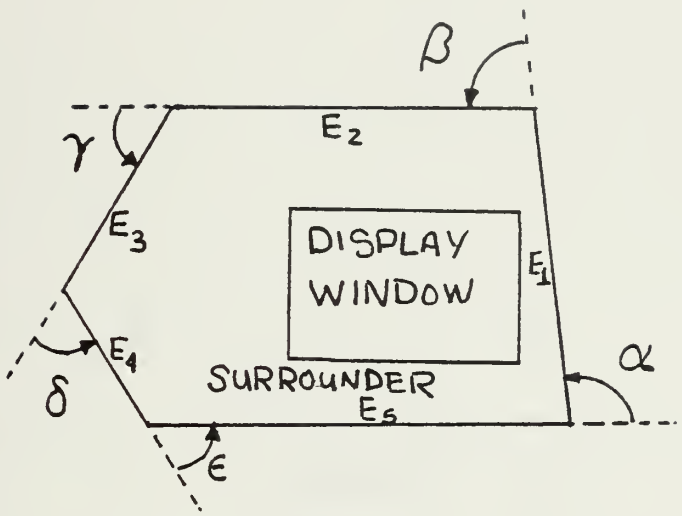


SURROUNDER



DISJOINT

Figure 15 - POLYGONAL CLASSIFICATIONS



$$\alpha + \beta + \gamma + \delta + \epsilon = 360^{\circ}$$

Figure 16 - EDGES'S ANGULAR COMPUTATION

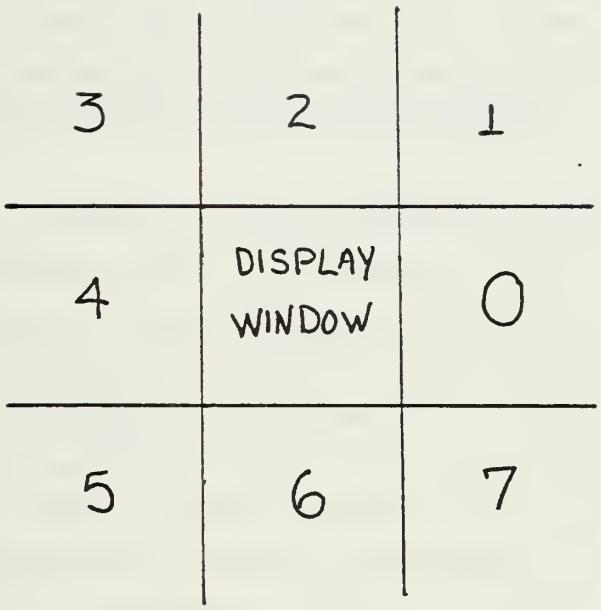


Figure 17 - DISPLAY AREA DIVISION AND CODING

The last angle computational problem occured when the magnitude was equal to four, as shown in Figure 18. The correct sign in this situation was not rendered by taking the difference of the two sector values. Counter-clockwise rotation or movement about a polygon's edges should have yeilded a positive result. The problem was resolved by selecting any point between the two endpoints which was not in either of the vertices' sectors, as shown in Figure 19. By dividing the edge at this point, the correct angle could be computed by summing the angles of these "two edges". This angular computation for an edge was determined in the clipping subroutine used by this proceedure.

The linked lists of the classified polygons were then passed to the Thinker. The surrounder list was processed first to determine which polygon was closest to the viewer by computing the depth of each of these polygons at the four window corners, as shown in Figure 20. Provided the closest polygon, called the hider, was not penetrated by another polygon, these four depths were used to determine if an intersector polygon was located completely in front of the hider within the confines of the display window. If an intersector was completely hidden from the viewer by the hider, it was removed from the list. If the final intersector list contained only one polygon, then that part of the polygon's edges which were inside the window were displayed. If the list contained more than one intersector, or if any intersector polygon penetrated the plane of the hider, the Thinker announced failure for that window. If the hider was penetrated by another surrounder, the Thinker announced failure before examining the intersector list. The penetration of one polygon by another was shown in Figure 21. Whether polygon B would be classified as an intersector or a surrounder penetrating polygon A would depend on the placement of the window. When the complexity

of the display could not be resolved and the size of the window had been reduced to the display device's smallest resolution, a dot was displayed at the window's lower, left corner. In this manner, the penetration of one polygon by another, which described a line, was displayed as an implied edge.

If failure was announced, and the size of the window was larger than the smallest resolution, the display window was divided into four equal windows and pushed onto the top of the stack. The Controller then selected the next window on the stack and passed it to the Looker. This entire process was summarized by the flow chart in Figure 22.

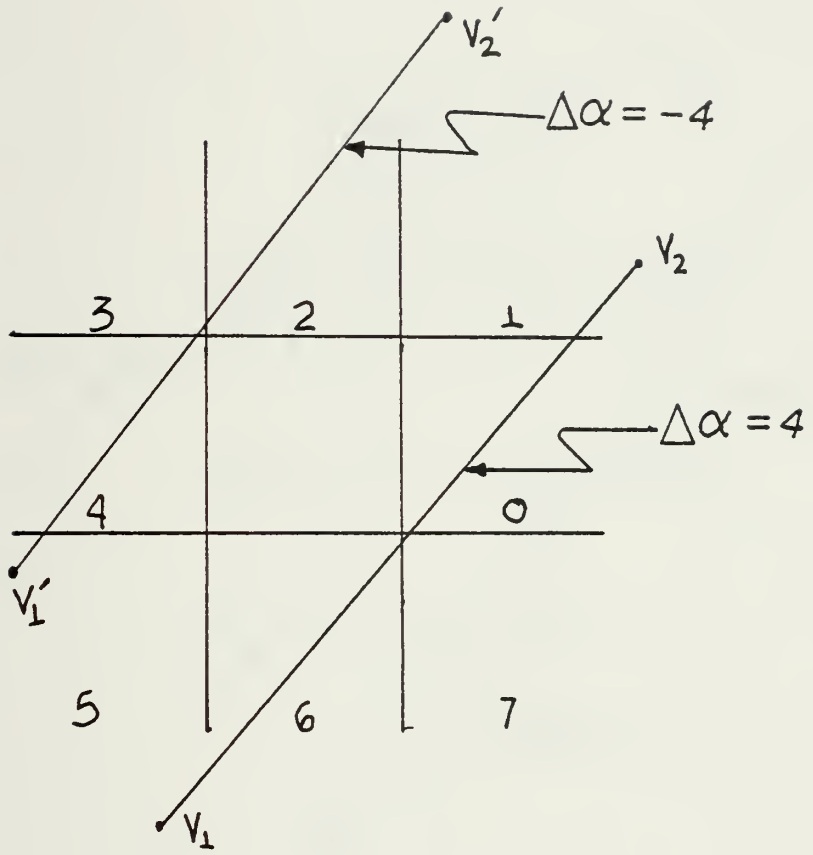


Figure 18 - ANGLE MAGNITUDE OF FOUR

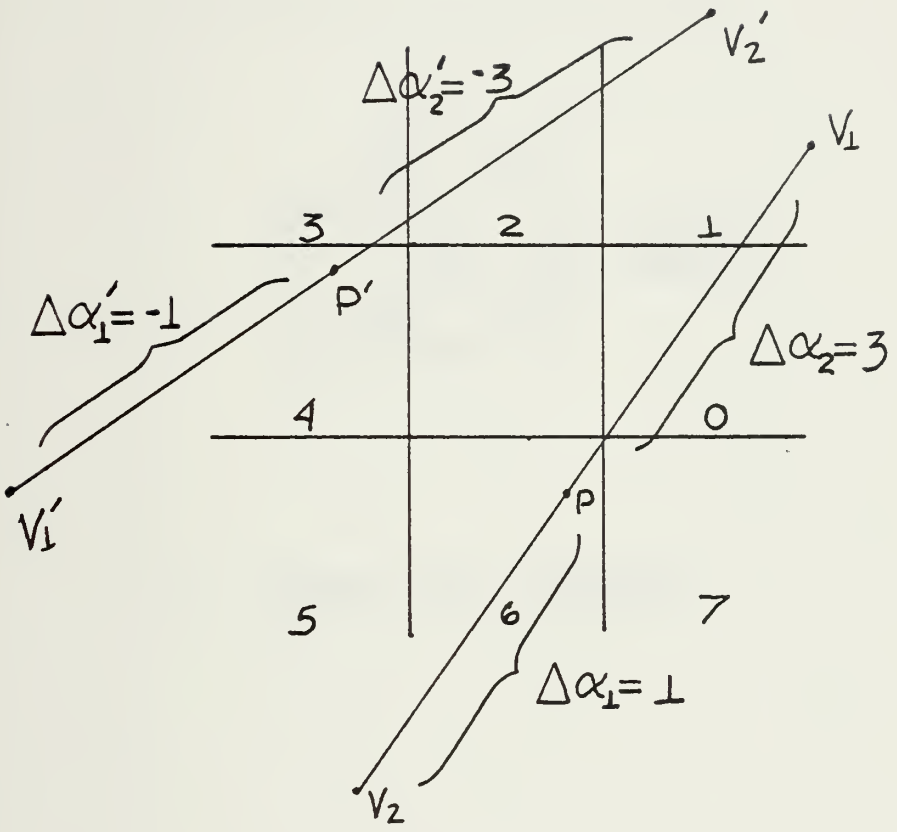


Figure 19 - EDGE DIVISION FOR PROPER ANGLE COMPUTATION

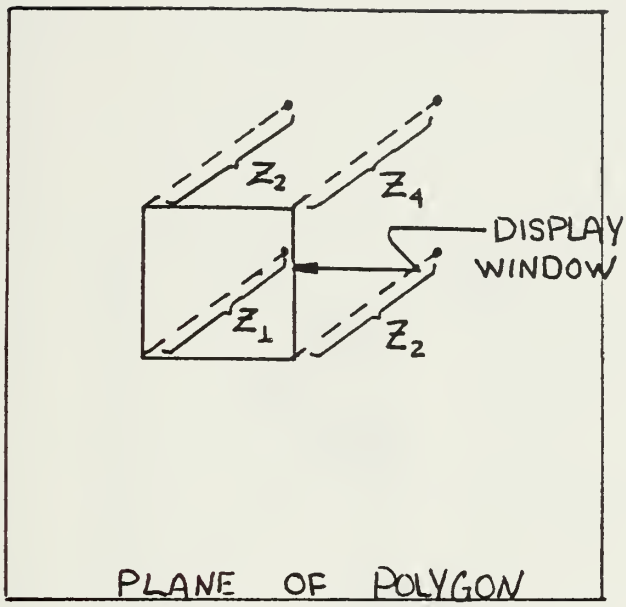


Figure 20 - DEPTH COMPUTATION OF POLYGON AT FOUR CORNERS OF WINDOW

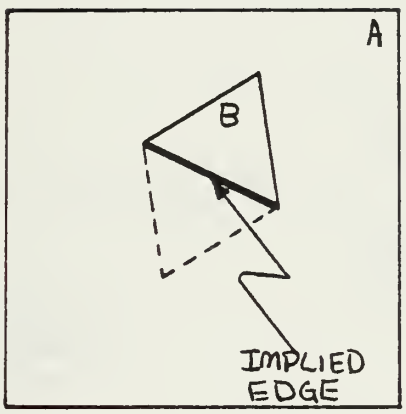


Figure 21 - PCLYGCNAL PENETRATION

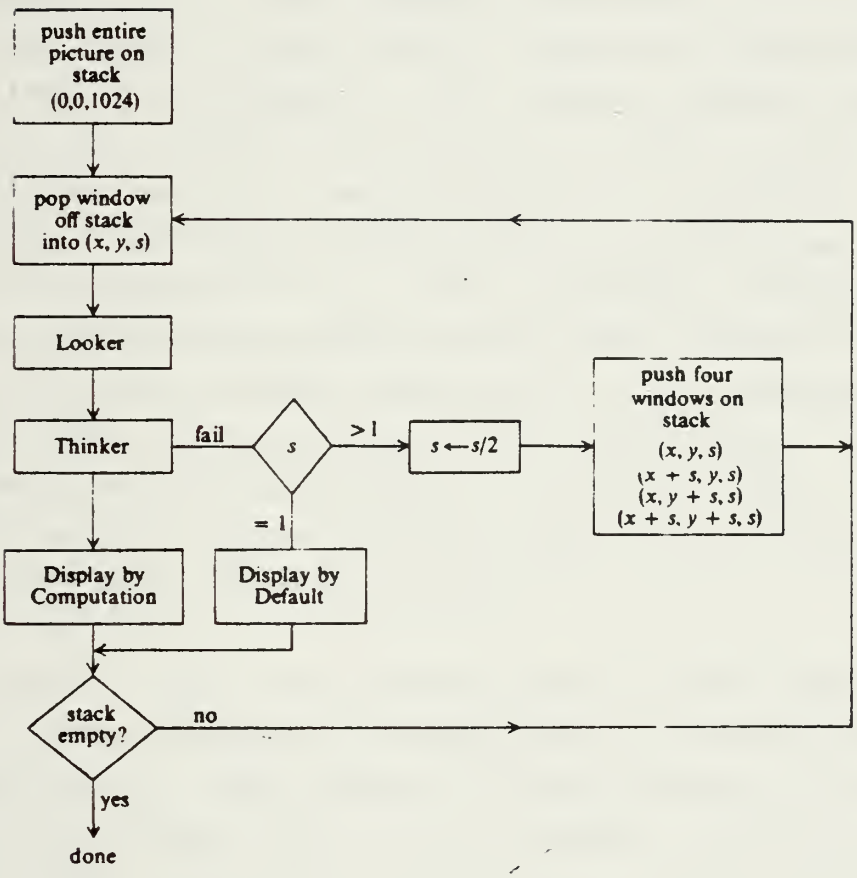


Figure 22 - HIDDEN LINE REMOVAL FLOW CHART

B. HIDDEN SURFACE REMOVAL

The algorithm utilized for this procedure was developed by G. S. Watkins at the University of Utah. The program was interpreted from a SAIL program presented in Ref. [1]. While the hidden line removal algorithm concentrated on linked lists of polygons, the hidden surface algorithm processed the display with lists of edges. Additionally, the vertices had to be expressed in clipped, screen coordinates. Although the user still generated the data for an image as stated in A., this procedure displayed the polygonal surfaces as a solid plane using shading or colors. Thus, a 3-D object should become much more realistic when displayed with shaded surfaces vice wire frames. While the algorithms presented previously in section III., can be utilized on any display device, this concept was developed specifically for raster scan CRT's.

A raster scan is a special type of CRT, which is very similar to the television in most homes. The vectored CRT's and the direct view storage tubes generate a display by pointing an electron beam to a desired location on the display screen and then moving it to any other screen location. This process illuminates the phosphorous screen to produce a single line segment. The typical television receives an analogue broadcast signal which generates a single horizontal line of the screen's image at a time. At the end of each line a horizontal sync pulse is received to move the electron beam down one line, and to the left-hand edge. When the last horizontal line has been displayed a vertical sync pulse moves the beam to the top, left-hand corner of the screen. Since the phosphorous screen remains illuminated a very short time, the image must be constantly

refreshed, typically at a rate of thirty times per second.

A raster scan display device receives its image (and refresh) information from random access memory (RAM) refresh planes where the image is stored as a sequence of individual bits. Each bit of a memory plane determines the illumination of a single element on one horizontal dislay line (also called a scan line or a raster). A picture element, called a pixel, is the smallest screen resolution size. The standard sixteen level grey shading requires four bits, one bit on four planes, to represent the shading of one pixel. Similar memory requirements are needed to display an image with sixteen colors. While the vectored CRT's have acheived resolutions on a display screen of 4096 lines with 4096 elements per line, the finest resolution available with raster scan devices is 1024 by 1024. Thus, a sixteen color, raster scan display with high resolution required four million bits of RAM. Because of this extensive memory requirement, the developement of this type of display device followed that of the small, lower cost electronic memory.

As shown in Figure 23, the intersection of the plane of a scan line with a polygon was a line segment. (Scan line k corresponds to the $Y_s = k$ plane.) This line segment's endpoints were defined by its Xleft, Zleft, Xright, and Zright values, which were the X_s and Z_s coordinates of the intesection of the scan line with two of the polygon's edges. The two, 2-D equations below were used to find the intersection of an edge with each scan line.

$$X_s = a\,Y_s + b \quad (1)$$

$$Z_s = c\,Y_s + d \quad (2)$$

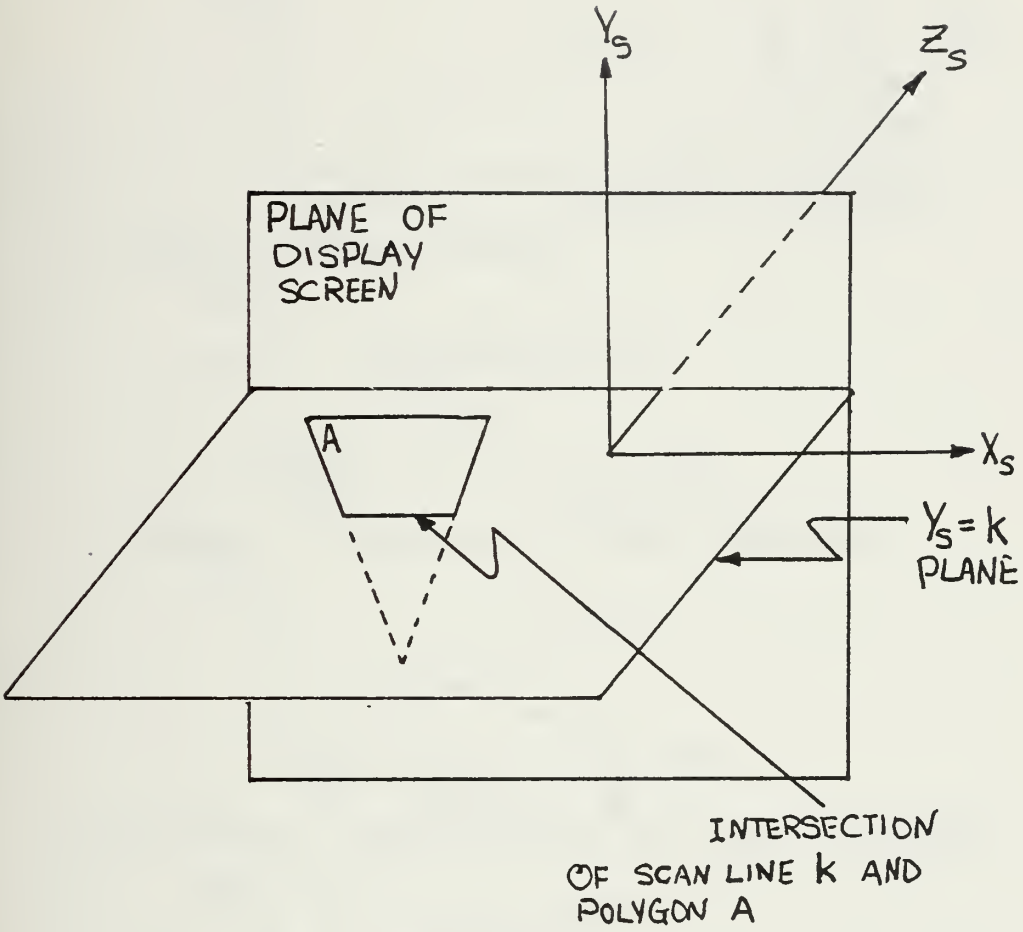
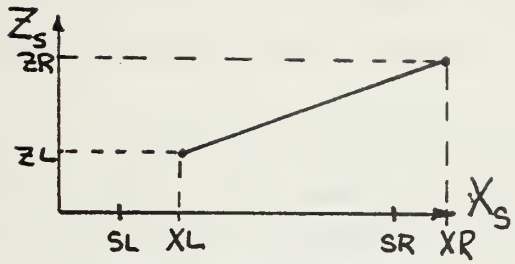
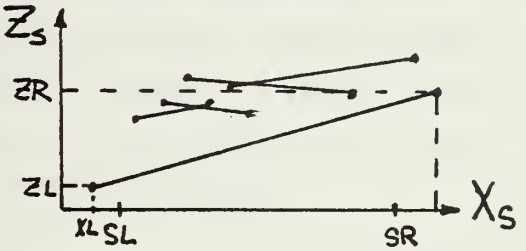


Figure 23 - SCAN LINE INTERSECTION OF A POLYGON

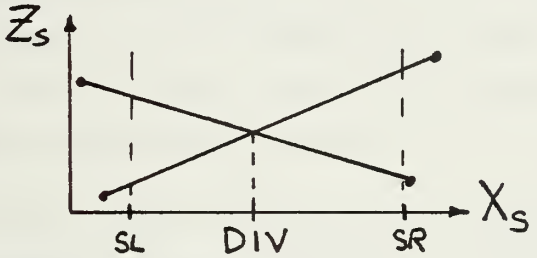


SL = SPAN LEFT
SR = SPAN RIGHT
XL = XLEFT
XR = XRIGHT
ZL = ZLEFT
ZR = ZRIGHT

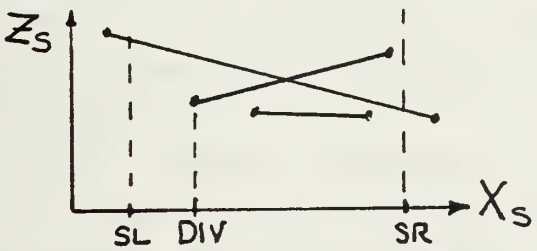
1. ONE SEGMENT IN SPAN



2. SPANNER HIDES ALL OTHER SEGMENTS



3. SIMPLE INTERSECTION OF TWO SPANNERS



4. TOO COMPLICATED - DIVIDE SPAN AT DIV

Figure 24 - SEGMENT CLASSIFICATION

The coefficients, a, b, c, and d, were quickly obtained using the equations for a 2-D line with the coordinates of the two vertices describing each edge.

With these equations the X and Z values of the intersection of an edge with scan line k+1 was the values at scan line k plus their respective slopes, a and c. Since each edge, and thus each segment, sustained only an incremental change between scans lines, the display was also assumed to remain constant between scan lines. This scan line coherence of the display was used to decrease the time required to process an image.

To process each scan line, it was necessary to divide the line into spans which could be more easily resolved. The content of a span could be categorized as shown in Figure 24 and as described below:

1. "The span contained only one segment.
2. One segment was closer to the viewer than all others and it was a spanner. A spanner was a segment where Xleft ≤ Span left and Xright ≥ Span right, as shown in Figure 24.
3. There was a simple intersection of the only two segments in the span, and both were spanners. This span was divided at the intersection into two spans and processed as in 1.
4. The display was too complicated in this span so it was divided at the left-most segment endpoint, or at the spans mid-point if there was no endpoint. The new spans were then processed." [7]

Since the lower, left-hand corner of the raster scan machine at the Naval Postgraduate School, the RAMTEK, was indexed as (0,0) each edge was ordered so that the index of

the vertex with the largest Y_s value was stored in EDGE(1,i). The integer value formed by truncating this vertice's Y_s value determined the first scan line that an edge would enter the display. A linked list of the indicies of the edges which entered on each scan line was stored in the array ENTLST(i). The index of the first edge to enter on scan line k was stored in YENTER(k).

As each edge entered the display, the X and Z values and the scan line coherence factors, the slopes for equations (1) and (2), were computed. Since the object of this algorithm was to display polygonal surfaces, the indices of the current segments of a polygon were linked in POLSEG(i). The segments were ordered by increasing X values of their left endpoint and the first segment's index was placed in SEGLST(p) for polygon p. A segment's index pointed to a block of storage which defined the endpoints X and Z values in the arrays XLEFT(i), ZLEFT(i), XRIGHT(i), and ZRIGHT(i) and their respective slopes DXLEFT(i), DZLEFT(i), DXRGHT(i), and DZRGHT(i). Integer arrays IYLEFT(i) and IYRGHT(i) were used to indicate when an edge, the source of one endpoint of a segment, was exiting the display.

To properly insert an entering edge into a polygon's segment lists required the comparison of the edge's X value for this scan line to the Xleft and Xright values of all of the active segments. If two edges entered on the same scan line at the same X_s coordinate, as shown in Figure 25, the edge with the largest slope was entered first. Thus, to enter the two edges on scan line k+1, between the two existing edges, a new block of storage was added for each entering edge. As each edge exited the scene, it was removed from its half of the storage block, as shown on scan

line k+2 in Figure 25. After all additions and deletions had been performed, the list of segments was sorted to conslidate storage. This entire process has been portrayed in Figure 26. As stated before, an edge usually separated two polygons. To eliminate redundant operaticns, the array P(2,i) was used to store the indices of the polygons ccmmon to edge i. When an edge entered the display and separated two polygons, its values of intersection were added to the blocks of stcrage for both segment lists.

This algcrithm was divided into the same three parts, the Thinker, the Looker, and the Contrcller, as the hidden line routine. The Looker compared all segments which intersected a span and developed sufficient information for the Thinker to process it. Provided the contents of a span satisfied the categories 1 through 3 above, the Thinker was able to generate the data required for displaying this span. If the infcrmaticn contained in the span was too complicated, it was divided by the Ccntroller. The sucessful scan line division points normally cccurred at the left-most endpoint of a segment. Since this division point's location on the next scan line can be predicted, it was stored and used to decrease the time required to process the entire image. The computation of the X and Z values of a segments endpoints and an updated, sorted list of segments was also performed by the Controller. This active list of segments was sorted in the Xsort lists, IXSLFT(i) and IXSRGT(i), by increasing X values. These twc lists provided the index cf the segment to the left and to the right of segment i by:

k = IXSLFT(i) : was the index of the segment to the left.

j = IXSRGT(i) : was the index of the segment tc the right.

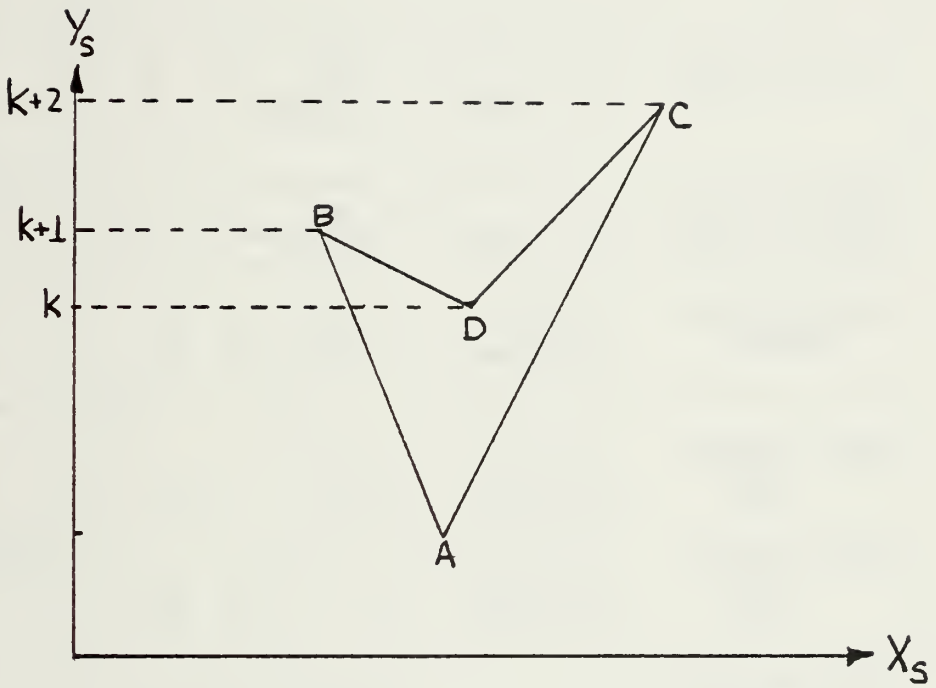


Figure 25 - ENTERING AND DEPARTING SEGMENTS

SCAN LINE	POLYGONAL SEGMENT STORAGE BLOCKS			
k		L_1 | R_1		ONE POLYGONAL SEGMENT
k+1	1.	L_1 | R_3	L_2 | —	ADD EDGE DB
	2.	L_1 | R_3	L_4 | -	L_2 | — ADD EDGE DC
	3.	L_1 | R_3	L_4 | R_2	CONSOLIDATE STORAGE
k+2	1.	— | R_3	L_4 | R_2	DELETE EDGE AB
	2.	— | —	L_4 | R_2	DELETE EDGE DB
	3.	L_4 | R_2		RETURN BLOCK TO FREE LIST

WHERE THE SUBSCRIPTS: 1 = EDGE AB
2 = EDGE AC
3 = EDGE BD
4 = EDGE CD

AND: L = LEFT AND R = RIGHT

Figure 26 - UPDATE OF SEGMENT BLOCK STORAGE

The last bookkeeping task performed by the Controller was to divide the list of segments into the following categories:

1. "SEGOUT- the right edge of the segment was contained in this span. (This segment did not appear in any span to the right of this one.)
2. SEGACT-the right edge of the segment extended beyond the right limit of this span." [8]

If this span was displayed, then the SEGOUT list did not need to be considered in subsequent spans and could be discarded (until the next scan line). The SEGACT list was automatically added to the next spans active segment lists. If the span failed to be processed, the two lists were combined and compared to the new span.

When the segment lists were passed to the Looker, the X and Z coordinates of the left-most and the right-most parts of a segment in this span were computed (see Figure 27 where the following terms are portrayed: sxleft, szleft, sxright, and szright). A box was constructed about the first segment examined in the X_s-Z_s plane which entirely surrounded that part of the segment which intersected the span. As each new segment was compared to the box, the box was enlarged to include it. If a segment completely hid the box from the viewpoint or if it was a spanner, then the count of the segments in the box was reset to one and the box was made to enclose only this segment. This was shown in Figure 28 along with the definitions of the box X and Z limits, bxleft, bzleft, bxright, and bzright.

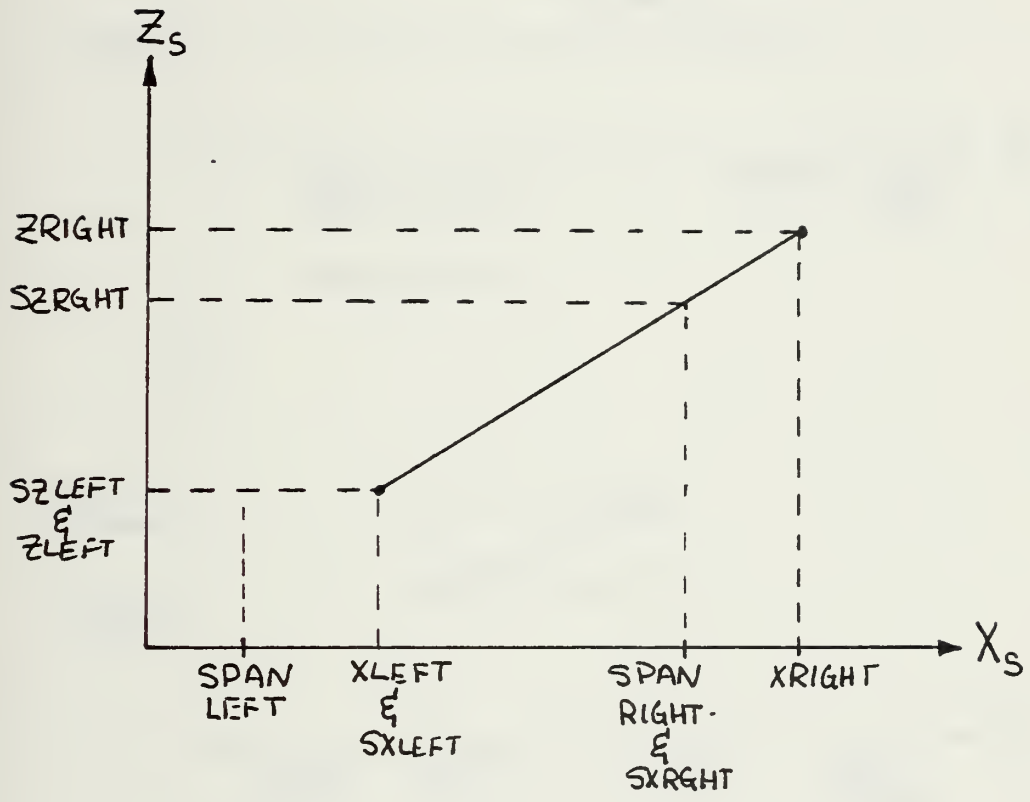


Figure 27 - SPAN DESCRIPTIVE TERMS FCR A SEGMENT

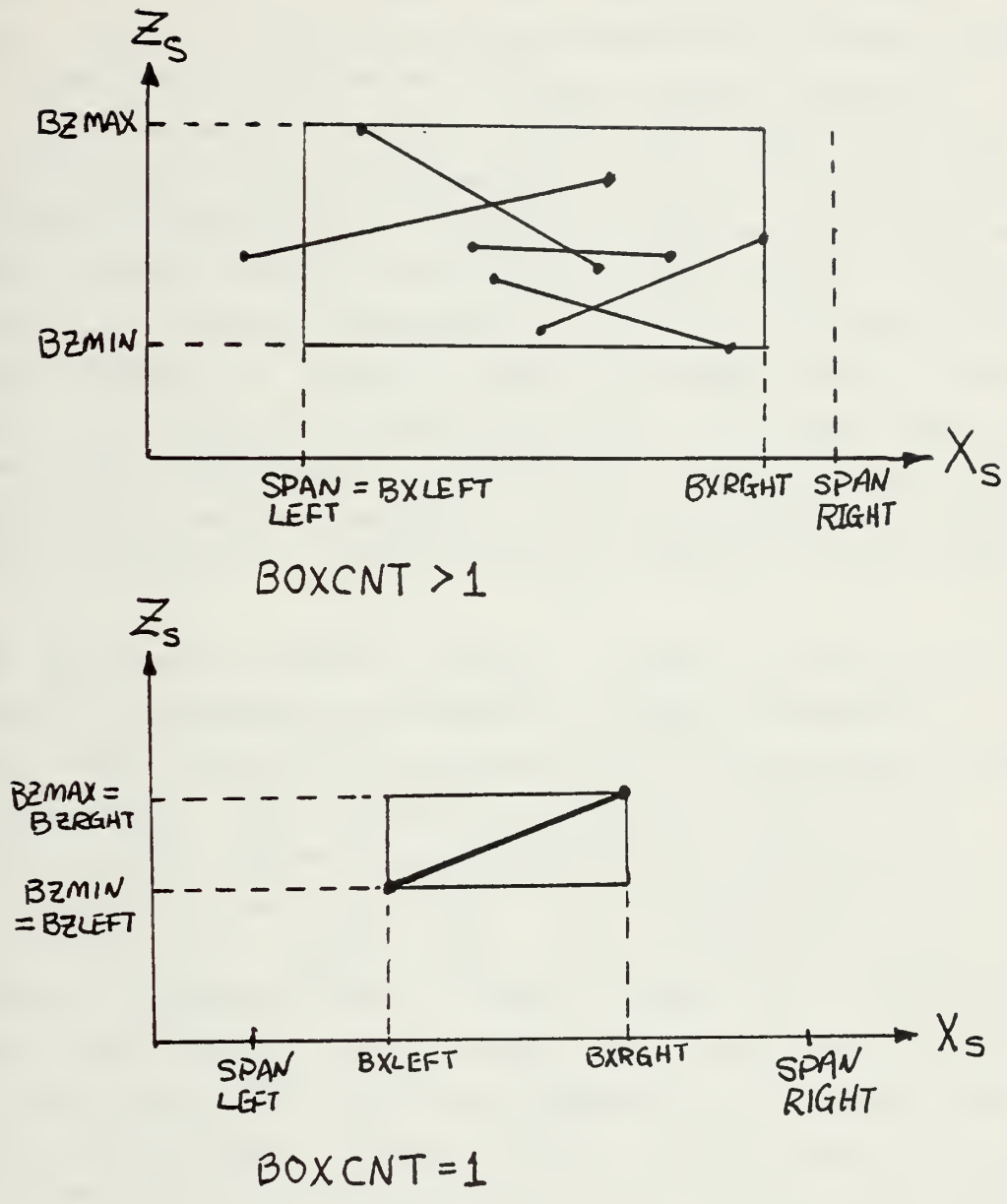
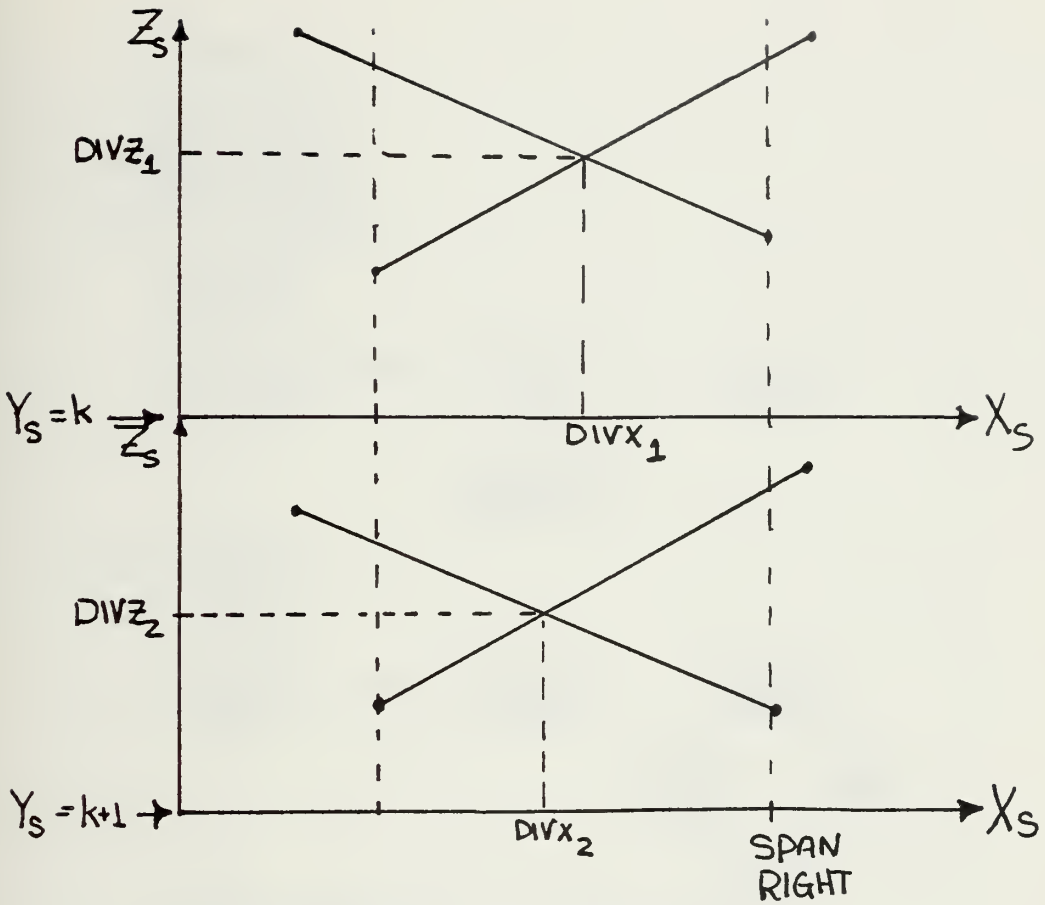


Figure 28 - SEGMENT BOX DEFINITION

The information passed to the Thinker was box count and box type. If the box count was zero, the Thinker did nothing and the Controller began processing the next span. A box count of one indicated a single segment existed in the span. The segment's sxleft, sxright and index were stored with the results of previous spans so that the entire scan line could be displayed at one time. When the box count was greater than one and the box type was equal to one, the span contained a simple intersection of two spanners. Both segments' sxleft and sxright values and their indices were stored with those of previous spans for this scan line. If the box count exceeded one and the box type equaled zero, the Thinker announced failure. The Controller then divided the span and began to process the left half of the old span.

An important element of all hidden surface or line elimination algorithms has been the proper display of an implied edge. When the intersection of two polygons caused an implied edge, this proceedure stored the value DIV, the division point caused by two intersecting spanners, on its first occurence (see Figure 29). The occurrence of this division on a second scan line provided sufficient information to compute its X and Z scan line coherence factors. The implied edge was then added to a dummy segment block and sorted with the other segments in the Xsort lists. However, this dummy segment block was not passed to the Looker, but was used to divide the scan line into the proper spans. When the division point due to an implied edge divided the same segment, the implied edge was discarded. A generalized flow chart was provided for this algorithm in Figure 30 to summarize this entire section.



SLOPE FOR EQUATION (1):

$$a = DIVX - DIVX_1$$

SLOPE FOR EQUATION (2):

$$c = DIVZ_2 - DIVZ_1$$

Figure 29 - IMPLIED EDGE GENERATION

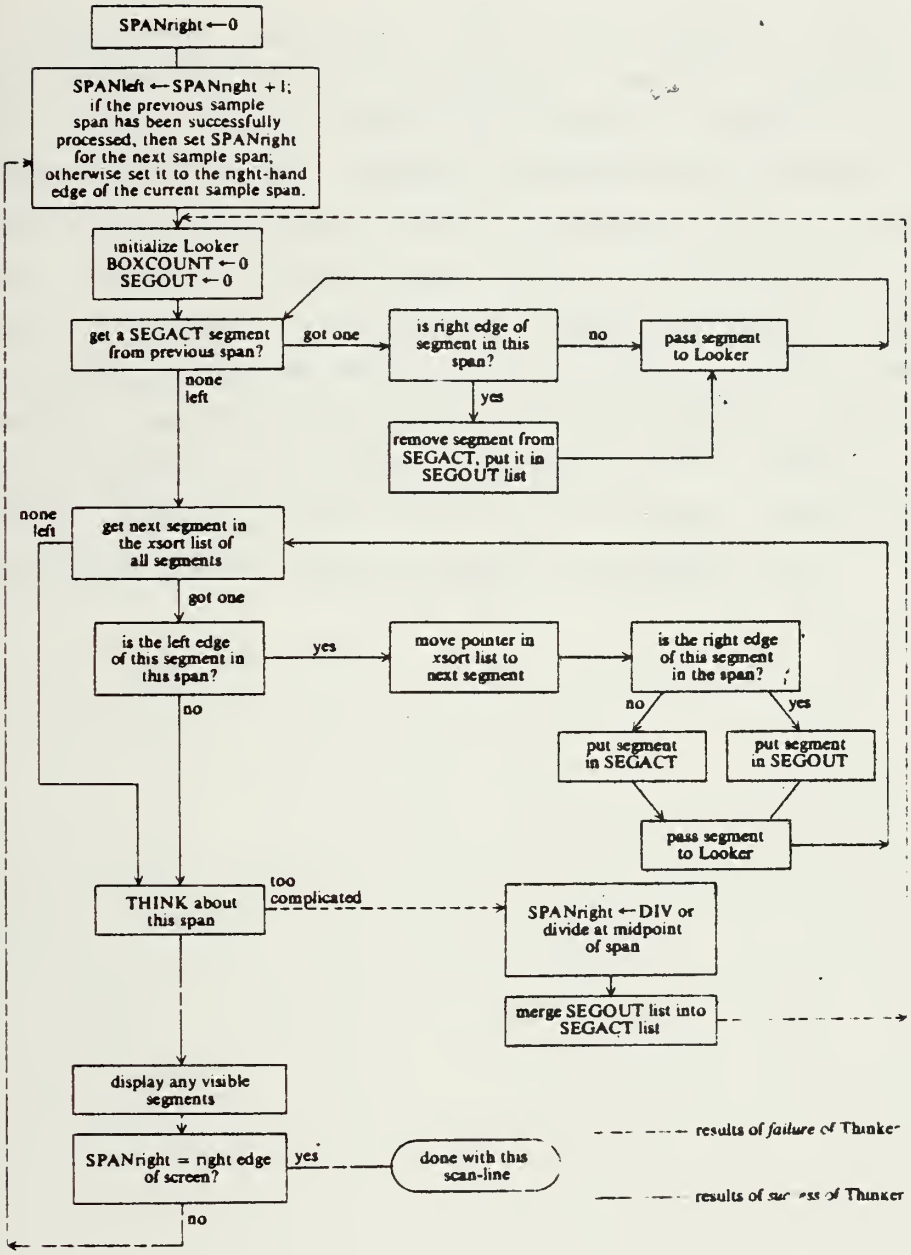

SPANright ← 0
SPANleft ← SPANright + 1; if the previous sample span has been successfully processed, then set SPANright for the next sample span; otherwise set it to the right-hand edge of the current sample span.
initialize Looker
BOXCOUNT ← 0
SEGOUT ← 0
get a SEGACT segment from previous span?
got one
is right edge of segment in this span?
no
pass segment to Looker
yes
remove segment from SEGACT, put it in SEGOUT list
none left
get next segment in the xsort list of all segments
got one
is the left edge of this segment in this span?
yes
move pointer in xsort list to next segment
is the right edge of this segment in the span?
no
yes
put segment in SEGACT
put segment in SEGOUT
pass segment to Looker
no
THINK about this span
too complicated
SPANright ← DIV or divide at midpoint of span
merge SEGOUT list into SEGACT list
display any visible segments
SPANright = right edge of screen?
yes
done with this scan-line
no
results of failure of Thinker
results of success of Thinker

Figure 30 - HIDDEN SURFACE REMOVAL FLOW CHART

I. IMAGE SHADING

An extremely important aspect of computer image realism was the generation of an appropriate shading algorithm. The realistic algorithms required complex scftware and thus, more computation time. Another alternative used has been to implement the algorithm with sophisticated and expensive hardware. Although this was one key element to a 3-D graphics language, a software implementation would have required half again as much reasearch and time. Therefcre, this aspect was left for future developement. The usage of colors or shading for object definition was input via the subroutine INITAL along with the cther image data (a complete description was included in Appendix A).

IV. HARDWARE AND SOFTWARE CONSIDERATIONS

The realistic implementation of a standard graphics language throughout a large organization would require that the majority of the software could be input to all host computers with minimal alterations required for specific display devices. Therefore, the primary intent was to reduce display device dependence to the fewest number of subroutines. The processing of a 3-D display in real time was another important consideration. In the following sections the algorithms in III. were divided into four groups and their implementition on the display devices was presented.

A. IMAGE DISPLAYED - ALL LINES SHOWN

The device dependent software which had to be utilized to display an image and the different screen or display area of each machine were the two non-portable aspects of this entire graphics package. Since the intent of this effort was to produce a language which appeared to be device independent to the user, the problem of various display areas was resolved in the image data input subroutine, INITAL. When more than one display device was supported by a host computer, the user had to select the appropriate display device number. INITAL then chose the correct line and element resolution and the location of the display area's center. The four devices used in this project were:

1. TEKTRONIX 4012- a direct view storage tube;

2. ADAGE AGT-10- a vectored CRT;
3. VERSATEC- a hard copy device which also had electrostatic shading capability;
4. RAMTEK- a raster scan CRT.

The format of the data input by the user was stated in the graphics language description preceeding INITAL. The required data included the image description as stated in III. A. and the following information:

1. a - the distance from the screen to the viewer;
2. b - the vertical dimension of the screen;
3. the x, y, and z coordinates of the viewpoint;
4. the select display device number;
5. the index of the color table to be utilized (when using the RAMTEK);
6. and the index of the color or shading for each polygon.

A complete explanation of the input data was provided in Appendix A with the program listing of INITAL.

To display the image as input without removing hidden lines, the user then had to call the subroutine DISPLY. DISPLY called the subroutines listed below.

1. RDYCLP- performed the object to clipping coordinate transformation on the vertices and stored the results in the arrays XS(i), YS(i), and ZS(i);
2. CLIP- clipped the image against the viewable display area;
3. SCRN- converted the clipping coordinates to screen coordinates.

The image could now be displayed by the selected device. First, each machine had to be initialized by a single device dependent subroutine call. Next, the line segments of each

edge, also called vectcrs, were drawn by another device dependent subroutine. Finally, all devices, except the RAMTEK, required a subroutine call which terminated the image. The ADAGE also required an image subroutine which developed a display list of vectors which was used to refresh the display screen. Thus, the device dependent portions for direct image viewing could typically be contained ir three subroutines and at most four when a display list was required. The display device was initialized by the subroutine INITAL and vectors, cr line segments, were drawn by DISPLY. The graphics software package was always terminated by a call tc FINISH, which when required called the device dependent subroutine to terminate the display.

The only calling parameter which was required for DISPLY was a two element integer array, IR(2), which was used tc:

1. display a single polyhedron by setting IR(1) = IR(2) = the index of the polyhedron;
2. display a group of consecutively input polyhera, where:
   IR(1) = the index of the first;
   and IR(2)=the index of the last polyheron;
3. cr display the entire image as input in INITAL by setting IR(1) = 31.

Except fcr the TEKTRONIX, the display cf a wire frame image by video graphic machines was performed in "real time". Real time was defined as a period cf time so short that the user could not detect a time lag between the programs execution and the complete drawing of the image. While the ccmputation of the image require an insignificant amount cf time for the host computer of the TEKTRONIX (an IBM-360), the vector display speed of this device was very slow. The VERSATEC was also a relatively slcw device, but

hard copy machines have not been expected tc produce "real time" displays.

The construction of this general graphics language did omit some specific device versatilities. The VERSATEC had the capabilities to re-define its display area, resolution size, and line thickness, and could produce shaded images. These capabilities could have been added, but were left for subsequent efforts due to time constraints.

E. INTERACTIVE SOFTWARE AND HARDWARE

The transformations for rotation, scaling, and translation provided the means for complete image movement in three dimensions. These procedures, which have been implemented with hardware at some installations, coupled with interrupt devices, such as alphanumeric keyboards, function switches, graphic tablets, joysticks, light pens, and track balls, provided a user with a complete interactive viewing capability. The subroutines written to rotate, scale, and translate a single polyhedron or the entire image were completely device independent.

The calling parameter IR(2), which was used by each of the three subroutines, was defined and utiilized as stated in A. above. Thus, new viewing aspects could be generated for a single polyhedron, a group of objects, or the entire displayable image. The usage of the scaling subroutine also required that the user provide the scale factors for the x, y, and z coordinates. The additional calling parameters required for image translation were the distances in the x, y, and z directions which the 3-D object was to be moved. The sign required for these distances was opposite to that of the standard velocity vector describing the objects

motion in this direction. Rotation calling parameters included the number specifying the axis and the angle of rotation. If an arbitrary axis was selected, the coordinates of two distinct points had to be passed into the subroutine also. The usage of these routines was defined in the comment section preceeding INITAL. The three software transformations, rotation, translation, and scaling, required a minute amount of computation time. Any combination of a few of these three called between INITAL and DISPLY could be performed in "real time".

Direct view storage tube display devices have an extremely limited interactive capability due to the method used to clear the screen. This device had a writing cathode which traced the image on a fine wire mesh which was located just behind the phosphorous screen. Initially, the entire wire mesh was negatively charged. Vectors drawn on the mesh by the writing cathode caused those line segments to become positively charged. These positively charged areas accelerated and passed the electrons emitted from a second cathode, which was issuing a "flood" of electrons to refresh the image on the phosphorous screen. To clear an image, a large positive pulse was applied to the wire mesh. This caused a large flash to spread across the screen. Since the flash disrupted any possible display for several seconds, the usage of this type of device for rapidly changing, interactive images was highly unrealistic.

Vectored CRT's have acheived an extremely high degree of resolution and support most interactive interrupt devices. With this type of display machine, the image on the phosphorous screen was refreshed by storing the entire image in a vector list. A highly complex, static image, one containing several thousand vectors, could cause the display to begin flickering. This type of CRT can produce only a limited number of vectors before the phosphorous illuminated

for the first vector begins to dim. Thus, shaded images, which would require many vectors for even a simple 3-D object, can not feasibly be produced using a vectored CRT. However, excellant 3-D graphs and extremely complex wire-framed objects have been visulized, using multiple colors, with these devices.

The ADAGE AGT-10 was an extremely versatile device and had an alphanumeric keyboard, function switches, function knobs, a joystick, and a light pen as interactive capabilities. Since this machine was operated in a stand-alone mode, these interactive interrupt devices were easily utilized through the users application program, except the light pen. The light pen could only be accessed and utilized in the image subroutine. This extremely, device dependent capability was not included in the graphics software, since its usage would have required a thrcugh knowledge of this machines software for even a simple application.

The RAMTEK's interactive devices consisted of an alphanumeric keyboard and a set of function switches which could be used to position a cursor. The cursor's screen coordinates were obtainable through device related software. The RAMTEK's host computer was a PDP-11, which primarily supported the software "C". Since the graphics software for the RAMTEK was written in C and there was no software interface written for Fortran IV, only those subroutines required to perform hidden surface elimination were translated to Fortran. Therefore, the interactive capability of this device was limited to input via a host computer's alphanumeric terminal. The resolution of this video graphics device was 240 lines with 640 elements per line. Its lack of vertical discrimination provided poor image continuity in this dimension. Since this was the only raster scan device available at this school, its utilization

was necessary to implement the hidden surface algorithm.

C. HIDDEN LINE REMOVAL

The hidden line removal algorithm described in Section III. was invcked by calling the subroutine REMOVE after INITAL and any desired interactive subrcutines. REMOVE called the subroutines RDYCLP, SCRN, WARNCK, and DISPL2. The calling parameter required by REMOVE was the integer array IR(2), which was described in A. Since this algcrithm clipped each edge against display windows to process an image, the vertices of the polyhderon to be displayed were passed tc WARNCK expressed in un-clipped screen coordinates. WARNCK contained the Lcoker, the Thinker, and the Contrcller described in III. As the display was processed, the X_s and Y_s coordinates of the two endpoints for each vector were stored in two arrays. This storage reduced the number of device dependent subroutines added to the graphics package by this algcrithm to cne, DISPL2. DISPL2 generated vectcrs for the display exactly as performed by DISPLY. The image subroutine used by DISPLY to created the vectcr list fcr the ADAGE was also used by this subroutine.

The algcrithm as presented in Sectien III. displayed each edge of a simple polyhedron, like a cube, by failure. Display by failure means that the Thinker was unable to resolve any display window and a dot was displayed when the window's size was reduced to the smallest screen resolutien. An edge was displayed as a line cf dots and was extremely grainy. This consistent failure had occurred because each edge was common to two polygcns. When the display window was reduced so that it contained only cne edge, the

intersector list still contained the indices of two polygons. The Thinker announced failure and the Controller divided the window. The computation time required to process and display a cube (which has a maximum of three viewable surfaces and nine viewable edges) exceeded twelve minutes of CPU time on the IBM-360. Additionally, almost 9,000 storage locations were required to store the endpoints of these single dot vectors.

To reduce the occurrence of display by failure, the intersector list was not rejected if only two polygons remained after comparison with the hider. When the list contained two indices, the number of edges which intersected the current display window was determined for both polygon's. Provided there was only one edge for both, the vertex indices of one polygon's edge were compared to those of the seconds to ensure it was the same edge. When this procedure found a common edge, the endpoints of the vector intersecting the window were stored. This addition to the Thinker reduced the CPU time to approximately 20 seconds and the storage requirements to about 300 locations.

Even though the display time had been reduced by a factor of thirty, this hidden line removal procedure was not even remotely acceptable for a real time display. While a more complex Thinker could possibly reduce the computation time, hardware implementation remains the only feasible method for a real time display using this algorithm. Additionally, the realism of wire-frame images, even with hidden lines removed, was marginal at best. As stated previously, shaded surfaces, or solid images, can not be displayed with this type of CRT. Therefore, realitic 3-D, "real time" computer graphics must be performed using a raster scan CRT with a hidden surface removal algorithm.

D. HIDDEN SURFACE REMOVAL

The hidden surface algorithm was implemented by calling the subroutine SURFAC after INITAL and any of the desired interactive procedures. As with all other procedures in this package, the hidden surfaces may be removed and the display created for one or a set of polyhedra or the entire image input to INITAL as determined by the two element integer array IR(2). The object coordinate data was transformed to clipped, screen coordinates by calling the subroutines RDYCLP, CLIP, SCRN, and SHOWIN. The remaining portion of this subroutine, SURFAC, was the Controller as described in Section III. The Looker and the Thinker were contained in the obviously named subroutines LOOKER and THINKER.

The hidden surface removal algorithm added two device dependant subroutines to the graphics software package, which were:

1. SHOWIN - constructed the desired color table and corrected the vertical coordinates;
2. SHOW - displayed each scan as its image was resolved.

The hidden line subroutines stored all of the image vectors until the entire display was resolved. However, even a single polyhedron would generate such a large list of vectors (or segments) to display solid surfaces that the storage requirements of a moderately complex scene would exceed realistic limits.

The RAMTEK, and most raster scan CRT's, could produce sixteen intensity levels for each of the three primary

colors. Thus, with all possible combinations of the shads of red, green, and blue, this device was able to display any 16 of the 4096 colors at one time. Since a shading algorithm was not implemnted, SHOWIN was used to construct a color table, containing sixteen colors, which was needed for the applications program presented in Section V. Additionally, the RAMTEK's vertical scan lines (240 lines total) were twice as wide as each horizontal element of resolution. To prevent this rectangular picture element from causing image elongation, the vertical dimensions, V_{sy} and V_{cy}, were doubled. This procedure caused the horizontal, X_s, coordinates to be clipped with Y_s values which were twice their actual size. Thus, the large disparity between vertical and horizontal resolution dimensions was not allowed to cause image elongation. SHOWIN was also used to divide all Y_s coordinates by two before the controller began processing scan lines. Had these coordinate values not been divided, each scan line displayed would have been resolved and written into RAM twice.

The hardware used to input the image data onto the memory planes by the RAMTEK was called a "vector generator". Although the "vector generators" used by current raster scan CRT's disply vectors at speeds only limited by the memory plane write times, the RAMTEK's generation of line segments was noticeably slow. Except for this slow vector display capability, this software algorithm developed shaded surfaces in "real time". Provided the user defined the shading or colors of each polygon, this type of display algorithm was shown to be a valuable tool for rapidly changing, realistic presentations.

V. A THREE-DIMESNIONAL GRAPHICS APPLICATION

In order to demcnstrate the capabilities of the 3-D Graphics Package an applicatien program was written, which was motivated by the display of the torpedo test area at Keyport. Each test area, which was irregularly shaped, could be described by one or more convex polyhedra. A pclyhedron's upper and lower surfaces represented the Puget Sound's air/water bcundry and its mud bottcm, respectively. Rotation, scaling, and translation of these polyhedra provided any desired viewing aspect. Since the display device at Keyport, the GENISCO GCT-3000, was a high resolutien (1024 by 1024) display device, the hidden surface algorith was selected to present a realistic 3-D display. However, the Range Safety Cfficer needed tc see inside the polyhedral approximatien of the torpedo test area not the closest polygonal surfaces. In view of this common type of display requirement, the algorithm was altered so that the hidden surfaces were displayed and the clcsest surfaces were deleted.

A real time display of torpedoes and other test vehicles in the range was required to prevent them from running aground cr leaving the test area. Hence, an algorithm was developed which detected the polygonal surface penetration by a vehicle. Obvicusly, detection of an accident after cccurrence wcuild be absurd. Thus, an actual implmentation of this type of display would require the ccnstruction of a safety factor by displaying surfaces which were within the actual test area boundries. This construction would be performed by the definition of the topedo test area when input to INITAL and actually did not alter the display

requirements.

The location of a vehicle in the test area was provided by acoustic line of bearings which were used to determine its position. For this application, the fix provided by the bearings was assumed to generate the x, y, and z object coordinates required for this graphics package. Since the acoustic information provided about the vehicle was not 3-D, the torpedo's location was defined as a single point. However, the actual display of the vehicle was 3-D which provided the correct relative direction of motion information (called target angle) to the observer. Although all surfaces of the torpedo were the same color, its hidden surfaces were removed for display since this algorithm required such minute computational processing time. Finally, the torpedo's track, its last five positions, was displayed as a line in the vehicles color.

A. PLANAR SURFACE PENETRATION

The theory used to determine if a point had penetrated one of the test area's boundries was originally conceived by L. G. Roberts to remove hidden lines from 3-D figures. It used the coefficients of the plane equation presented in Section III. G. "These coefficients, in the vector form [a b c d], were also the expression for a homogeneous vector normal to the plane (homogeneous refered to the representation of a 3-D point as a 1 by 4 vector, where d was an arbitrary scale factor). If the dot product of this normal and a vector in the viewing direction was positive, then the polygon was a back face of the polyhedron and thus, non-viewable. Face normals were also used to compute shading parameters. " [9]

Additionally, if two points were on the same side of the plane of a polygon, the dot product of either point with the normal vector would have the same sign. Since concave polyhedra would allow two interior points to be on different sides of a plane determined by a polygon, only convex polyhedra were utilized. Two adjacent convex polyhedra could have at most one common polygonal face. Since penetration of this common surface by a vehicle would have falsely indicated danger, these faces were eliminated. If a vehicle penetrated a test area boundry, its color was changed to red.

E. HIDDEN SURFACES DISPLAYED

To present the interior of a 3-D object, it was necessary to eliminate the normally viewable surfaces. The hidden surface algorithm, which originally eliminated back planes, was easily modified to display the hidden surfaces. Only the Looker subroutine had to be altered. The Looker in Section III. compared all segments which intersected a span in order to locate one segment which was closest to the viewer and hid all other segments in the span. This subroutine was altered to search for the farthest segment from the viewer which hid all other segments from a viewpoint located on the other side of the origin (on the same viewing axis), as shown in Figure 31.

Since the majority of this image was static, the torpedo test area only required processing to display the hidden surfaces initially and when the viewing aspect was changed. However, any moving image presented a special problem on this type of display device. To project the concept of motion on a raster scan CRT, it was not sufficient to simply translate or rotate and then display the new image, because

the original figure was written in the memory planes and would be displayed until replaced. With the GENISCO, it is possible to sample the memory planes defining the color of any one pixel, but the RAMTEK, which was designed and built ten years before the GCT-3000, did not have that capability. To delete an image of a torpedo with the GENISCO, one method would be to sample the two colors displayed on the screen at the two endpoints of each line segment which defined this figure. Provided colors at these two points were the same and the vehicles image was small, this color could be stored for this segment. It could then be used to restore the original background when the torpedo's position changed. However, the time required for these background color computations could be more than that to re-process and display the entire image again. Without this option, using the RAMTEK, the static image was input to the hidden surface display algorithm and re-drawn each time the torpedo's location changed. The torpedo's 3-D form and its track where then displayed at their new screen locations.

To provide the maximum display flexibility, it was necessary to provide a means to both remove and display hidden surfaces. Thus, an additional calling parameter, ILOOK, was added to the subroutine SURFAC (the Controller). This parameter was used to enable the Controller to select the correct Looker subroutine, where:

1. ILOCK = 1, called LOOKER which removed hidden surfaces;
2. and ILOOK =2, called LOOKR[1] which displayed the back polygonal planes.

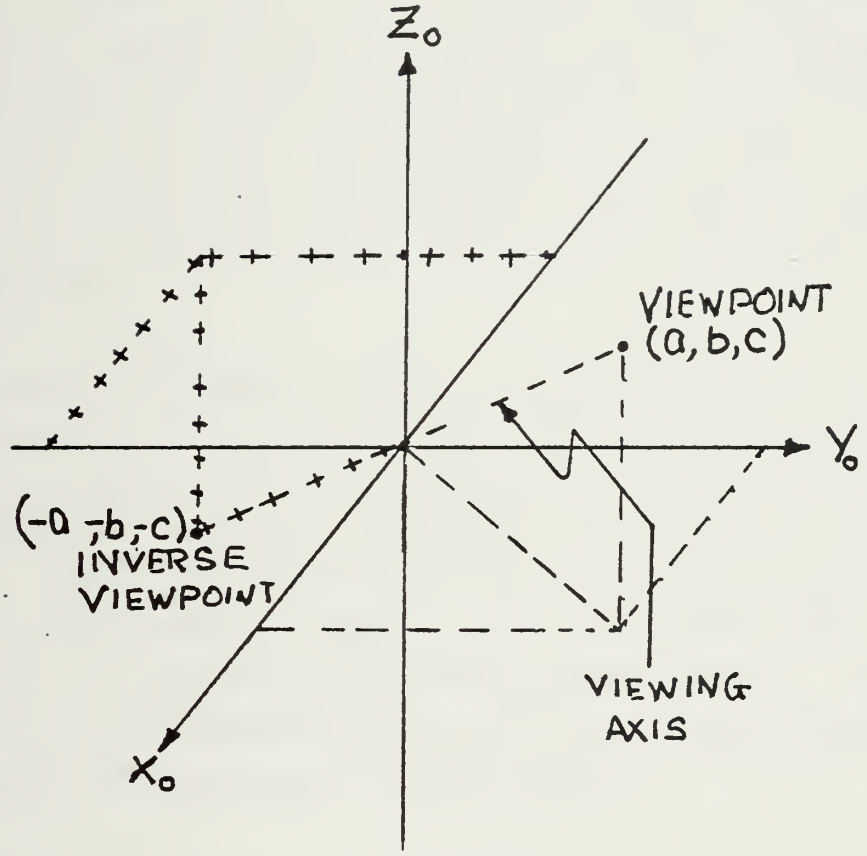


Figure 31 - VIEWPOINT REQUIRED TO DISPLAY THE BACK SURFACES

C. TORPEDO TEST AREA SIMULATION

The torpedo test area utilized for this application program was constructed with three, box type polyhedra. The two end boxes had five polygonal surfaces and the middle had four. The test area, which was shown in Figure 32 as a wire-frame image, had all surfaces displayed as light blue, except the bottom which was light brown. The torpedoes were colored black.

Interactive input to this program provided new torpedo positions in x, y, and z object coordinates, which simulated periodic acoustic fixes and subsequent display update. Any two, distinct vehicle locations described its direction vector. When a new position was input, a new direction vector was formed between this and the last torpedo location. To compute the correct target aspect, these two direction vectors were used to find the angle through which the vehicle had rotated, as shown below:

$$\cos\theta = (\underline{a}^{T}\ \underline{b}) \ / \ (||\underline{a}||\ ||\underline{b}||) \ ,$$

where $\underline{a}$ and $\underline{b}$ were the two direction vectors and the $||\underline{a}||$ operation represented the magnitude of vector $\underline{a}$.

The vehicle's image was then rotated about the axis which was normal to both vectors. This normal vector was found by forming the cross product of the two direction vectors. The two points used to specify this arbitrary axis required by the subroutine ROTATE were the vehicles present position, (x y z 1), and a point found using the normal vector. These two operations were summarized algebraically as :

$\underline{N} = \underline{a} \otimes \underline{b}$, where $\otimes$ represents the operation of vector cross product.

and

$(x' \; y' \; z' \; 1) = (x \; y \; z \; 1) + \underline{N}$

Thus, the correct vehicle aspect was obtained by translation of the image to (x y z 1) and rotation about the arbitrary axis specified above through the angle theta.

To realistically describe a 3-D object by planar polygonal mosaics was difficult if the object was composed of curved surfaces like a torpedo. Its cigar shape was basically represented by an octagonal cylinder. The rounded nose of the torpedo was roughly approximated by reducing the diameter of the cylinder. The smaller diameter of a torpedo's tail was exaggerated by reducing the cylinder's diameter to a point. An approximation of a propeller was attached to this point. As shown by the wire-frame image in Figure 33, even a crude approximation of such a complex surface required many polygons. The actual display of this image required the definition of 27 polygons, 58 edges, and 33 vertices. While the actual numerical values were not important, they showed that the storage requirements increased rapidly with the complexity of the surfaces which were to be displayed. Simple geometric figures, such as buildings, required little storage and the effort of approximation was minimal. The realistic approximation of complex surfaces required a large amount of storage and quite alot of artistic talent. Finally, these numerical values indicated, as observed with nearly all 3-D figures, that the memory allocations required to store edge definitions were usually double that for any other image discriptor.

The applications program, the polygonal penetration

subroutines, LOOKR1 (the modified Looker subroutine) and their respective flow charts were included in Appendix B.

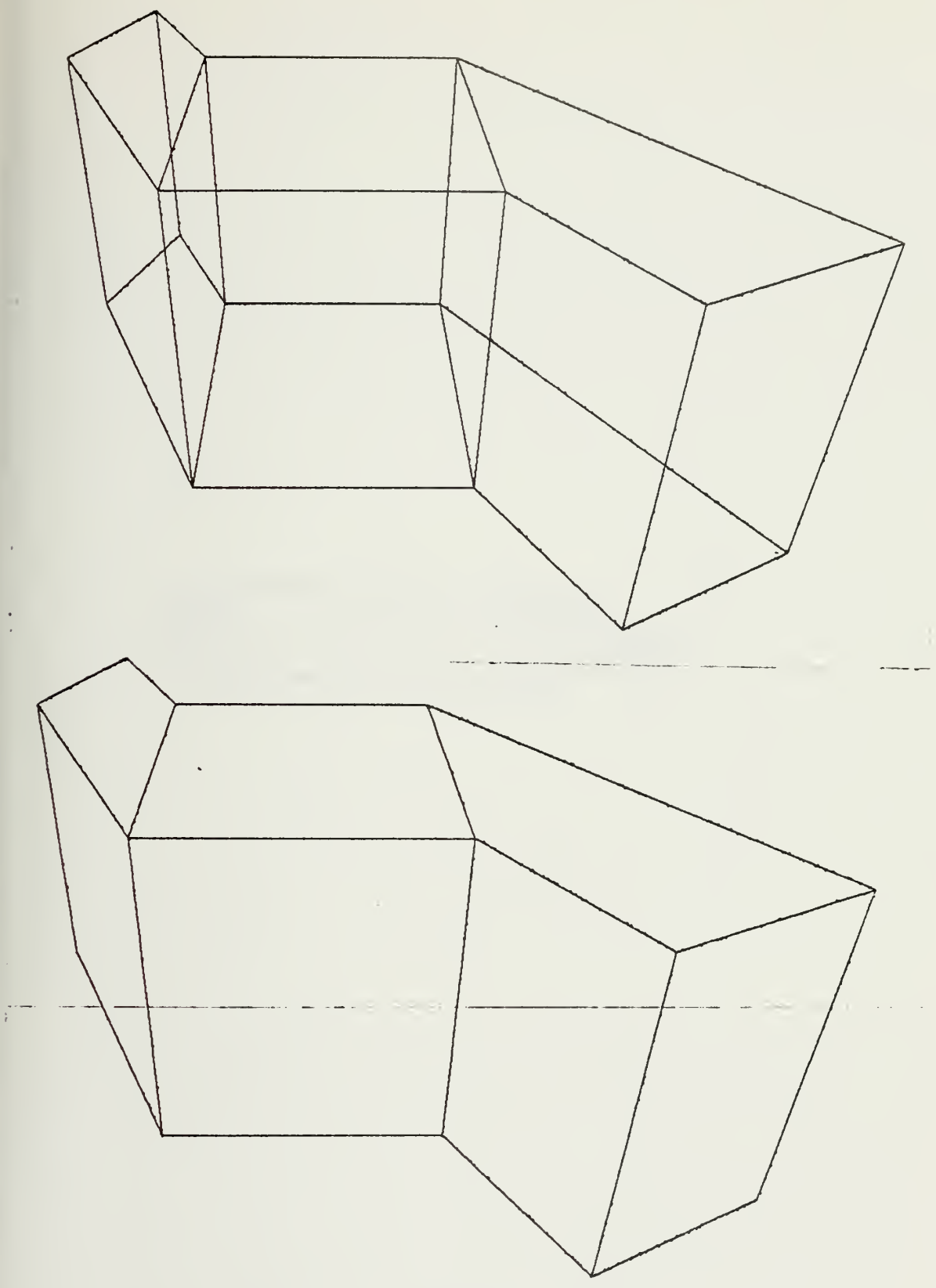


Figure 32 - TORPEDO TEST AREA

Figure 33 - TORPEDO APPROXIMATION

I. CONCLUSIONS

The Three-Dimensional Graphics Package contained in Appendix A did not include two important aspects of computer generated graphics, an image shading procedure and 2-D and 3-D graphs. Both items are important aspects of graphical presentations and should be included in a complete package. They were items left for future research due to the length of time required to develop the present graphical software.

The intent of this effort was to provide a portable computer graphics software package. To a very real extent, this goal was accomplished. However, the development software language, Fortran IV, was not entirely portable. In fact, a program which removed hidden lines on the IBM-360 failed on the PDP-11 because of one its fortran idiosyncrasies. Additionally, an abnormal amount of time was expended attempting to input data via a file on each new computer. It would make more sense for an organization to really standardize the fortran supported by all of its main computers before a portable graphics software was implemented.

Fortran as the development language for this graphics package proved to be quite efficient with one exception. If Fortran IV had the binary operations common to languages like C and SAIL, the graphics software could have been simplified. While no time comparisons between two languages were attempted, the excellent, "real time", results acheived with the hidden surface algorithm indicated that Fortran was highly acceptable as the development language. Fortran may not be the best language available for graphics, but it is

the only universally supported and accepted scftware.

The 3-D graphics package, as presented in Appendices A and B, prcvide a user with the ability to present any cbject on any selected display device which is supported through Fortran. Its portability was demonstrated cn four distinct types of display devices through the interface of three different host computers. There were only seven subroutines that contained statements which were device dependent. One of these, SHOWIN would not have been required if the raster scan CRT available had had square picture elements. These subroutines all call device procedures which performed the same task, but had different names. Each cf the device related subroutines caused the device to generate a line segment (cr vectcr) on the display surface. Usually, the display machines were built around a 16-Bit display processor. The primary task of the device dependent procedures was a data conversion interface between the host and the display computer. Therefore, if a large organization utilized a "standard" Fortran and required that all interface subroutines utilize standard names, this Three-Dimensional Graphics Software could be made completely portable. The obvious advantages of this software installation would be a great reduction in software and personnel training costs.

APPENDIX A

THREE-DIMENSIONAL GRAPHICS SUBROUTINES AND FLOW CHARTS

Each subroutine was listed with a very brief explanation of its function in the comment section preceeding the program. A specfic flow chart followed each program. When the contents of a program's flow chart could not be presented on a single page, a generalized flow chart was listed first. It was followed by specific charts which amplified all blocks of the flow chart that were marked with a circled number in a lower corner, such as:

The subroutines were divided into three groups, Display a Wire-Frame Image, Remove Hidden Lines, and Hidden Surfaces. The list of variables defined on pages 9 through 16 were used throughout the programs and the flow charts.

1. Display a Wire-Frame Image

This group of subroutines included those to input the image data, the coordinate system transformations, the image clipping procedure, and the display subroutine for wire-frame images. Additionally, the interactive subroutines to rotate, scale, and translate an image were listed with their flow charts. The subroutines which were used to multiply matirces were included, but flow charts were not drawn due to their simplicity and lack of

applicability to the graphics scftware effcrt.

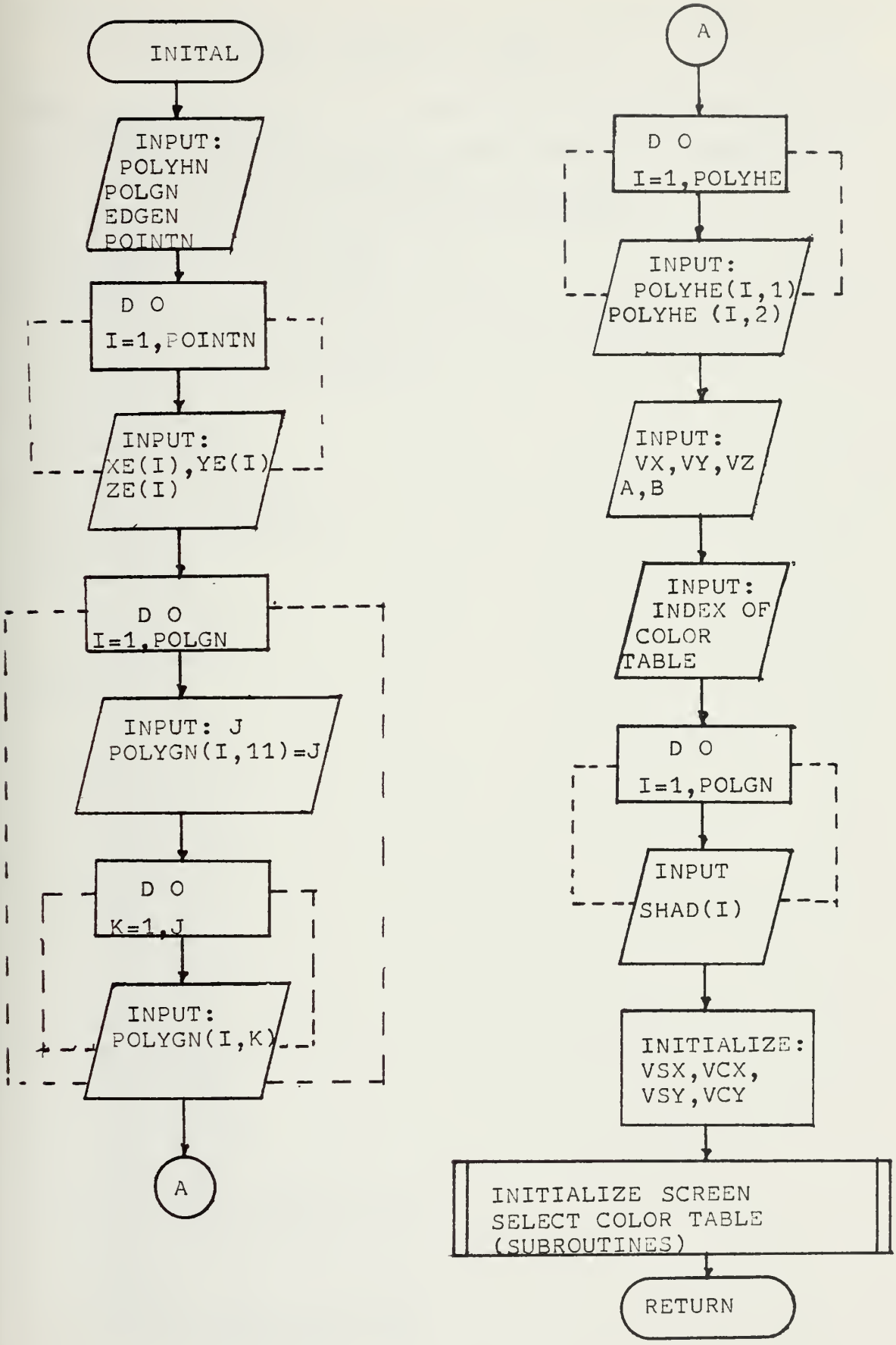
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     THE THREE DIMENSIONAL GRAPHICS PACKAGE CONSISTS OF THE FOLLOWING
C     USER CALLABLE SUBROUTINES:
C        1. INITAL- INPUTS ALL DATA WHICH DEFINES THE IMAGE;
C        2. TRANSL- TRANSLATES THE ENTIRE IMAGE OR A SINGLE POLYHEDRON
C        3. SCALE- SCALES THE ENTIRE IMAGE OR A SINGLE POLYHEDRON;
C        4. ROTATE- ROTATES THE ENTIRE IMAGE OR A SINGLE POLYHEDRON;
C        5. DISPLY- DISPLAYS THE IMAGE WITH ALL LINES DRAWN;
C        6. REMOVE- DISPLAYS THE IMAGE AFTER REMOVING ALL HIDDEN LINES
C        7. SURFACE- DISPLAYS THE OBJECT AS AN IMAGE WITH SOLID
C           SURFACES AFTER REMOVING THE HIDDEN OR BACK SURFACES.
C
C        THREE DIMENSIONAL IMAGES MUST BE CONSTRUCTED OF A SERIES OF
C     POLYHEDRONS (MAXIMUM OF 10). EACH POLYHEDRON MUST BE CONSTRUCTED
C     OF A SET OF CONNECTED POLYGONS (MAXIMUM OF 30 POLYGONS TOTAL).
C     EACH POLYGON IS DESCRIBED BY A SET OF 10 OR LESS CONNECTED EDGES
C     (THE MAXIMUM TOTAL NUMBER OF EDGES IS 60.)  EACH EDGE IS DEFINED
C     BY TWO DISTINCT POINTS.  EACH POINT (MAXIMUM OF 120) IS DEFINED
C     BY ITS X, Y, AND Z OBJECT COORDINATES.  (WHERE THE OBJECT
C     COORDINATE SYSTEM IS A THREE DIMENSIONAL, RIGHT HANDED SYSTEM
C     WHICH MAY USE ANY UNIT OF MEASUREMENT.)
C
C        TO USE THIS PACKAGE THE FOLLOWING DATA MUST BE SUPPLIED BY
C     CALLING SUBROUTINE INITAL:
C        A. USING A 4I4 FORMAT INPUT ,IN THIS ORDER:
C             (1) THE NUMBER OF POLYHEDRONS
C             (2) THE NUMBER OF POLYGONS
C             (3) THE NUMBER OF EDGES
C             (4) THE NUMBER OF POINTS
C        B. USING A 4G10.3 FORMAT INPUT THE X, Y, AND Z OBJECT
C             COORDINATES FOR EACH POINT.  THE POINTS ARE INDEXED,
C             CONSECTIVELY, AS THEY ARE INPUT.
C        C. USING A 2I4 FORMAT INPUT THE INDICES FOR THE TWO POINTS
C             WHICH DESCRIBE EACH EDGE.  THE EDGES ARE INDEXED,
C             CONSECUTIVELY, AS THEY ARE INPUT.
C        D. TO DESCRIBE EACH POLYGON INPUT, IN THIS ORDER:
C             (1) USING AN I4 FORMAT, THE NUMBER OF EDGES WHICH
C              DESCRIBE THIS POLYGON.
C             (2) USING A 10I4 FORMAT, THE INDEX NUMBERS OF THE EDGES
C              WHICH DESCRIBE THIS POLYGON.
C           THE POLYGONS ARE CONSECUTIVELY INDEXED AS THEY ARE INPUT.
C           TO REDUCE STORAGE SPACE ALL OF THE POLYGONS WHICH DETERMINE
C           A POLYHEDRON MUST BE INPUT CONSECUTIVELY.
C        E. TO DESCRIBE EACH POLYHEDRON INPUT, USING A 2I4 FORMAT, THE
C             INDEX NUMBERS OF THE FIRST AND LAST POLYGONS
C             WHICH DESCRIBE THIS POLYHEDRON.
C        F. USING A 3G10.3 FORMAT INPUT THE OBJECT COORDINATES
C             X, Y, AND Z, OF THE POINT FROM WHICH THE IMAGE IS TO BE
C             VIEWED.  THE VIEW POINT MUST BE EXTERIOR FROM ALL
C             POLYHEDRONS.
C        G. USING A 2G10.3 FORMAT INPUT THE DISTANCE FROM THE VIEWING
C             SCREEN THAT THE DISPLAY IS TO BE VIEWED AND THE VERTICAL
C             SIZE OF THE SCREEN.  THESE TWO MEASUREMENTS MUST USE THE
C             SAME UNITS.
C        H. USING AN I4 FORMAT INPUT THE INDEX OF THE COLOR TABLE TO
C           BE USED TO CONSTRUCT THE IMAGE.
C        I. USING AN I4 FORMAT INPUT THE INDEX NUMBER OF THE COLOR FOR
C           EACH POLYGON.  IF THE IMAGE IS TO BE DISPLAYED AS A WIRE-
C           FRAME (BY CALLING DISPLY OR REMOVE) THE ENTIRE IMAGE WILL
C           BE DRAWN USING THE COLOR OF THE FIRST POLYGON.
```

```
C
C       TO SHOW THE IMAGE WITHOUT REMOVING HIDDEN LINES CALL
C     SUBROUTINE DISPLY AFTER INITAL.  DISPLY HAS NO CALLING
C     PARAMETERS.  TO DISPLAY THE IMAGE AFTER REMOVING ALL HIDDEN
C     LINES, CALL SUBROUTINE REMOVE (NO CALLING PARAMETERS) AFTER
C     INITAL.
C
C       TO USE THE SUBROUTINES ROTATE, SCALE, AND TRANSL THE FOLLOWING
C     PARAMETERS MUST BE SPECIFIED VIA THE USER'S PROGRAM:
C        A. ROTATE(IR,IAXIS,P1,P2,THETA), WHERE:
C             1.IR- IS A TWO ELEMENT ARRAY WHICH DEFINES WHICH PART
C               OF THE IMAGE IS TO BE ROTATED, AS FOLLOWS:
C                  (A) TO ROTATE A SINGLE POLYHEDRON SET IR(1)=IR(2)=
C                    THE POLYHEDRON'S INDEX;
C                  (B) TO ROTATE SEVERAL CONSECUTIVELY INDEXED
C                    POLYHEDRONS SET:
C                    IR(1) = THE INDEX OF THE FIRST POLYHEDRON;
C                    IR(2) = THE INDEX OF THE LAST POLYHEDRON;
C                  (C) TO ROTATE THE ENTIRE IMAGE SET IR(1) = 31.
C             2. IAXIS DETERMINES THE AXIS ABOUT WHICH THE POLYHEDRON
C               OR ENTIRE IMAGE IS TO ROTATE, AS INDICATED:
C                  (A) IAXIS=0 - ROTATE ABOUT OBJECT X AXIS
C                  (B) IAXIS=1 - ROTATE ABOUT OBJECT Y AXIS
C                  (C) IAXIS=2 - ROTATE ABOUT OBJECT Z AXIS
C                  (D) IAXIS=3 - ROTATE ABOUT AN ARBITRARY AXIS WHICH
C                    MUST BE SPECIFIED USING THE ARRAYS P1 AND P2.
C             3. P1(3) IS AN ARRAY WHICH CONTAINS THE X, Y, AND Z
C               (OBJECT) COORDINATES OF ONE POINT ON THE ARBITRARY AXIS
C               OF ROTATION, WHERE:
C                  (A) P1(1) IS THE X VALUE
C                  (B) P1(2) IS THE Y VALUE
C                  (C) P1(3) IS THE Z VALUE.
C             4. P2(3) IS AN ARRAY CONTAINING THE X, Y, AND Z
C               COORDINATES OF ANY OTHER DIFFERENT POINT ON THE
C               ARBITRARY AXIS, WHERE P2(1), P2(2), AND P2(3) ARE USED
C               FOR X, Y, AND Z COORDINATES, RESPECTIVELY.
C             5. THETA IS THE ANGLE IN DEGREES THROUGH WHICH THE
C               POLYHEDRON OR THE ENTIRE IMAGE IS TO BE ROTATED.
C        B. SCALE(IR,S), WHERE:
C             1. IR IS AN INTEGER ARRAY USED TO DETERMINE WHICH PART
C               OF THE IMAGE IS TO BE SCALED AS DEFINED ABOVE FOR
C               ROTATE.
C             2. S(3) IS AN ARRAY WITH:
C                  (A) S(1) IS THE SCALE FACTOR FOR THE X COORDINATES.
C                  (B) S(2) IS THE SCALE FACTOR FOR THE Y COORDINATES.
C                  (C) S(3) IS THE SCALE FACTOR FOR THE Z COORDINATES.
C        C. TRANSL(IR,T), WHERE:
C             1. IR IS AN INTEGER ARRAY USED TO DETERMINE WHICH PART
C               OF THE IMAGE IS TO BE SCALED AS DEFINED ABOVE IN
C               ROTATE.
C             2. T(3) IS AN ARRAY WITH:
C                  (A) T(1) IS THE X DISTANCE TO TRANSLATE THE
C                    POLYHEDRON'S IMAGE.
C                  (B) T(2) IS THE Y DIST. TO TRANSLATE THE
C                    POLYHEDRON'S IMAGE.
C                  (C) T(3) IS THE Z DIST. TO TRANSLATE THE
C                    POLYHEDRON'S IMAGE.
C
C       THESE THREE SUBROUTINES MUST BE CALLED AFTER INITAL AND BEFORE
C     EITHER DISPLY OR REMOVE, INITIALLY. AFTER THE IMAGE IS INPUT
C     THE SUBROUTINE INITAL NEED NEVER BE CALLED AGAIN.  THUS, TO
```

```
C     CHANGE THE DISPLAY THE THREE SUBROUTINES ABOVE MAY BE CALLED IN
C     ANY SEQUENCE AFTER WHICH EITHER DISPLY OR REMOVE MUST BE CALLED.
C
C        THE CALLING PARAMETERS REQUIRED FOR DISPLY, REMOVE, AND
C      SURFACE ARE:
C          1. IR IS A TWO ELEMENT ARRAY WHICH DETERMINES WHICH
C            PART OF THE IMAGE IS TO BE DISPLAYED, AND IS USED
C            AS DEFINED ABOVE FOR ROTATE.
C          2. FOR THE SUBROUTINE SURFAC ONLY THE PARAMETER ILOOK
C            IS ALSO REQUIRED WHICH IS USED TO DETERMINE WHICH SURFACES
C            ARE TO BE DISPLAYED AND WHICH ARE TO BE REMOVED:
C              1. ILOOK=1 REMOVES THE HIDDEN SURFACES;
C              2. ILOOK=2 DISPLAYS THE BACK OR HIDDEN SURFACES,
C                WHICH ALLOWS THE USER TO SEE INSIDE THE POLYHEDRONS.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     INITAL: INPUTS THE IMAGE DATA
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE INITAL
      COMMON /AA/POLYHE,POLYHN
      COMMON /AB/POLYGN,POLGN,SHAD
      COMMON /AC/EDGE1,EDGE2,EDGEN
      COMMON /AAA/XE(120),YE(120),ZE(120),POINTN
      COMMON /CC/VX,VY,VZ,A,B,CX,CY,CZ
      COMMON /JJ/VSX,VSY,VCX,VCY
      INTEGER POLYHE(10,2),POLYGN(60,11),EDGE1(100),EDGE2(100),
     &POLYHN,POLGN,EDGEN,POINTN,RAMTEK,SCREEN,COLORT,SHAD(60)
     OPEN(UNIT=1,NAME='FORT01',TYPE='OLD',READONLY)
      READ(1,1) POLYHN,POLGN,EDGEN,POINTN
      READ(1,2) ((XE(I),YE(I),ZE(I)),I=1,POINTN)
      READ(1,3) ((EDGE1(I),EDGE2(I)),I=1,EDGEN)
      DO 100 I=1,POLGN
         READ(1,4) J
         POLYGN(I,11)=J
         READ(1,5) (POLYGN(I,K),K=1,J)
  100 CONTINUE
      READ(1,3) ((POLYHE(I,1),POLYHE(I,2)),I=1,POLYHN)
      READ(1,2) VX,VY,VZ
      READ(1,2) CX,CY,CZ
      READ(1,6) A,B
      READ(1,4)ICOLOR
      READ(1,4)(SHAD(I),I=1,POLGN)
      VSX=320.0
      VSY=240.0
      VCX=320.0
      VCY=240.0
      X1=0.0
      Y1=0.0
      X2=2.0*VSX
      Y2=VSY
      I=RAMTEK(IBB)
      IF(I.LT.0)WRITE(6,8)
      I=SCREEN(X1,Y1,X2,Y2)
      IF(I.LT.0)WRITE(6,9)
      I=COLORT(ICOLOR)
      IF(I.LT.0)WRITE(6,10)
      RETURN
```

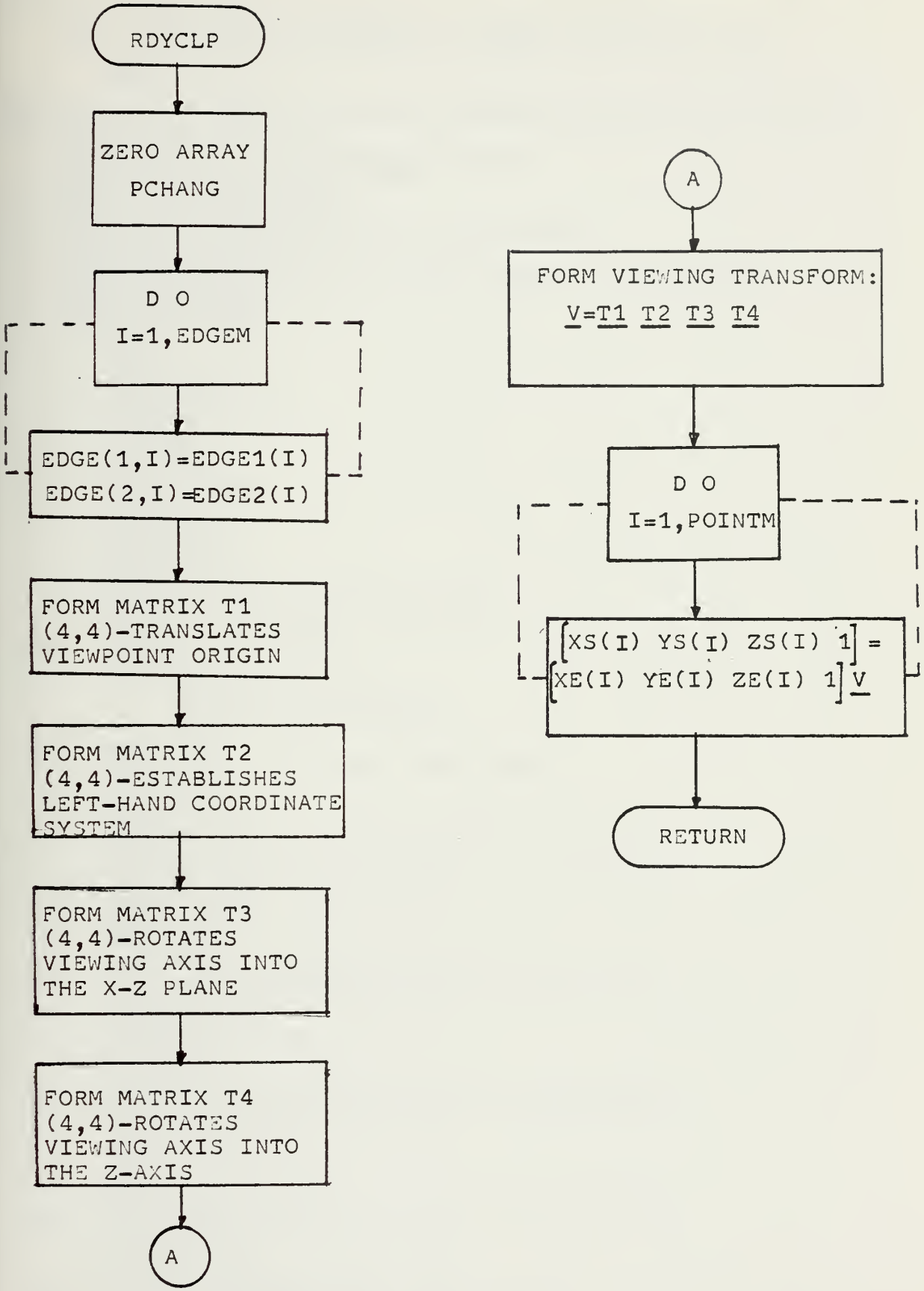
```
 1 FORMAT(4I4)
 2 FORMAT(3G10.3)
 3 FORMAT(2I4)
 4 FORMAT(I4)
 5 FORMAT(10I4)
 6 FORMAT(2G10.3)

 7 FORMAT(I1)
 8 FORMAT(2X,'THE RAMTEK DEVICE WOULD NOT OPEN')
 9 FORMAT(2X,'THE FUNCTION SCREEN FAILED')
10 FORMAT(2X,'THE FUNCTION COLORT FAILED')
   END
```


INITAL
INPUT:
POLYHN
POLGN
EDGEN
POINTN
D O
I=1,POINTN
INPUT:
XE(I),YE(I)
ZE(I)
D O
I=1,POLGN
INPUT: J
POLYGN(I,11)=J
D O
K=1,J
INPUT:
POLYGN(I,K)
A
A
D O
I=1,POLYHE
INPUT:
POLYHE(I,1)
POLYHE (I,2)
INPUT:
VX,VY,VZ
A,B
INPUT:
INDEX OF
COLOR
TABLE
D O
I=1,POLGN
INPUT
SHAD(I)
INITIALIZE:
VSX,VCX,
VSY,VCY
INITIALIZE SCREEN
SELECT COLOR TABLE
(SUBROUTINES)
RETURN

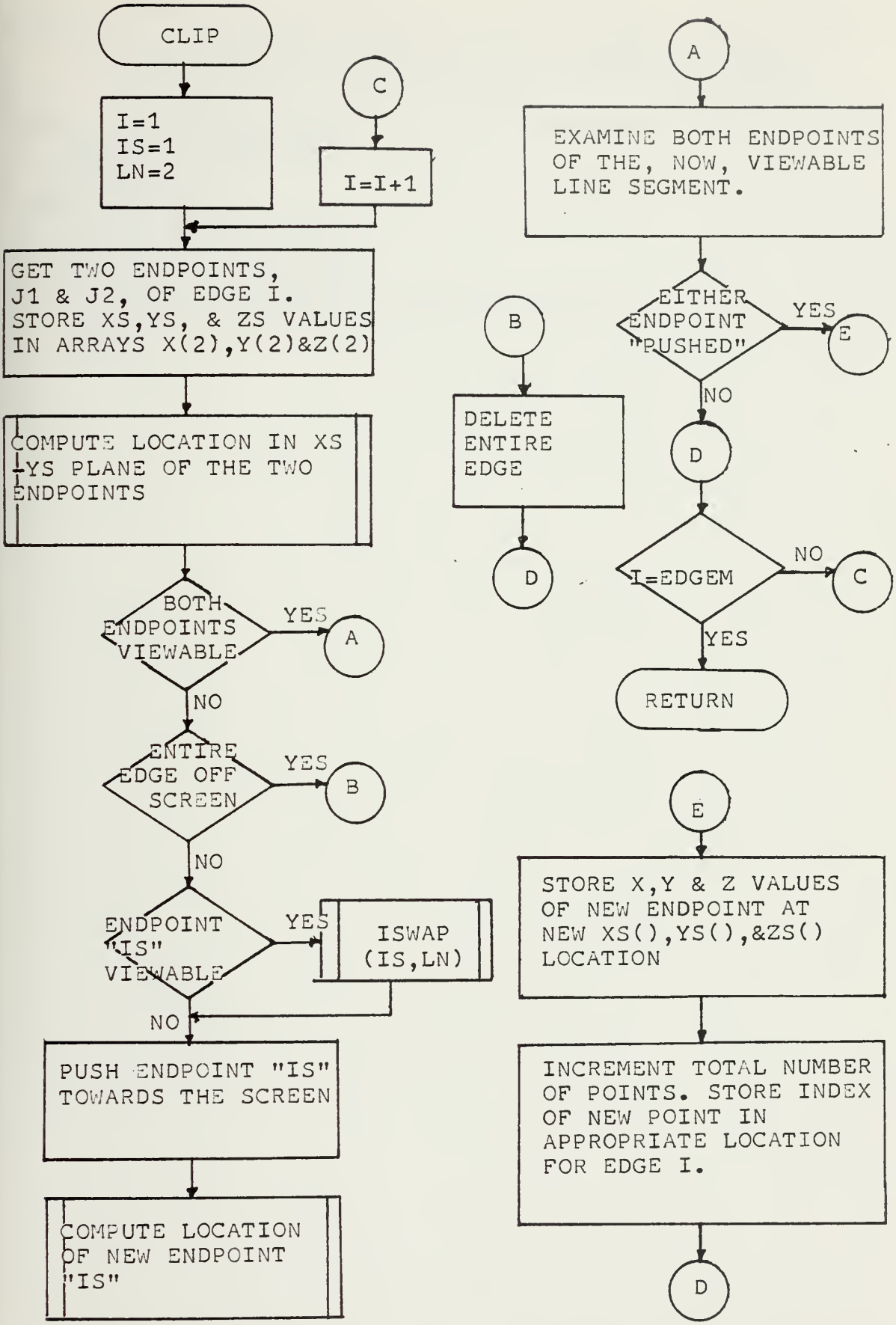
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     RDYCLP: TRANSFORMS THE OBJECT COORDINATES TO EYE COORDINATES
C        AND READYS THE IMAGE FOR CLIPPING
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE RDYCLP
      INTEGER POINTN,POINTM,EDGEN,EDGEM,EDGE(2,200),EDGE1(100)
      INTEGER PCHANG(200),EDGE2(100)
      COMMON /AAA/XE(120),YE(120),ZE(120),POINTN
      COMMON /AAB/ XS(120),YS(120),ZS(120),POINTM
      COMMON /AC/ EDGE1,EDGE2,EDGEN
      COMMON /AAD/ EDGE,EDGEM
      COMMON /CC/ VX,VY,VZ,A,B,CX,CY,CZ
      COMMON /FF/PCHANG,THETA
      COMMON /JJ/VSX,VSY,VCX,VCY
      DIMENSION V(4,4),T1(4,4),T2(4,4),T3(4,4),T4(4,4),RN(4,4)
      DIMENSION TNEW(4),TEMP(4)
      DATA V,T1,T2,T3,T4,RN/96*0.0/
      DO 30 I=1,120
   30 PCHANG(I)=0
      POINTM=POINTN
      EDGEM=EDGEN
      DO 160 I=1,EDGEN
      EDGE(1,I)=EDGE1(I)
      EDGE(2,I)=EDGE2(I)
  160 CONTINUE
      DO 136 I=1,4
      T1(I,I)=1.0
      T3(I,I)=1.0
      T4(I,I)=1.0
      RN(I,I)=1.0
  136 CONTINUE
      T1(4,1)=-VX
      T1(4,2)=-VY
      T1(4,3)=-VZ
      T2(1,1)=-1.0
      T2(2,3)=-1.0
      T2(3,2)=1.0
      T2(4,4)=1.0
      SQ=VX*VX+VY*VY
      SQQ=SQRT(SQ)
      CS=VY/SQQ
      SS=VX/SQQ
      T3(1,1)=CS
      T3(1,3)=SS
      T3(3,1)=-SS
      T3(3,3)=CS
      SQ=SQ+VZ*VZ
      SRQ=SQRT(SQ)
      CSS=SQQ/SRQ
      SSS=VZ/SRQ
      T4(2,2)=CSS
      T4(2,3)=-SSS
      T4(3,2)=SSS
      T4(3,3)=CSS
      CALL GMPRD(T1,T2,V,4,4,4)
      CALL GMPRD(T3,T4,T1,4,4,4)
      CALL GMPRD(V,T1,T2,4,4,4)
      SX=1.0
      SY=1.0
      IF(VSX.GT.VSY) SX=VSY/VSX
      IF(VSX.LT.VSY) SY=VSX/VSY
```

```
      RN(1,1)=A/B*SX
      RN(2,2)=A/B*SY
      CALL GMPRD(T2,RN,V,4,4,4)
      DO 137 I=1,POINTN
         TEMP(1)=XE(I)
         TEMP(2)=YE(I)
         TEMP(3)=ZE(I)
         TEMP(4)=1.0
         CALL GPRD(TEMP,V,TNEW)
         XS(I)=TNEW(1)
         YS(I)=TNEW(2)
         ZS(I)=TNEW(3)
  137 CONTINUE
      RETURN
      END
```

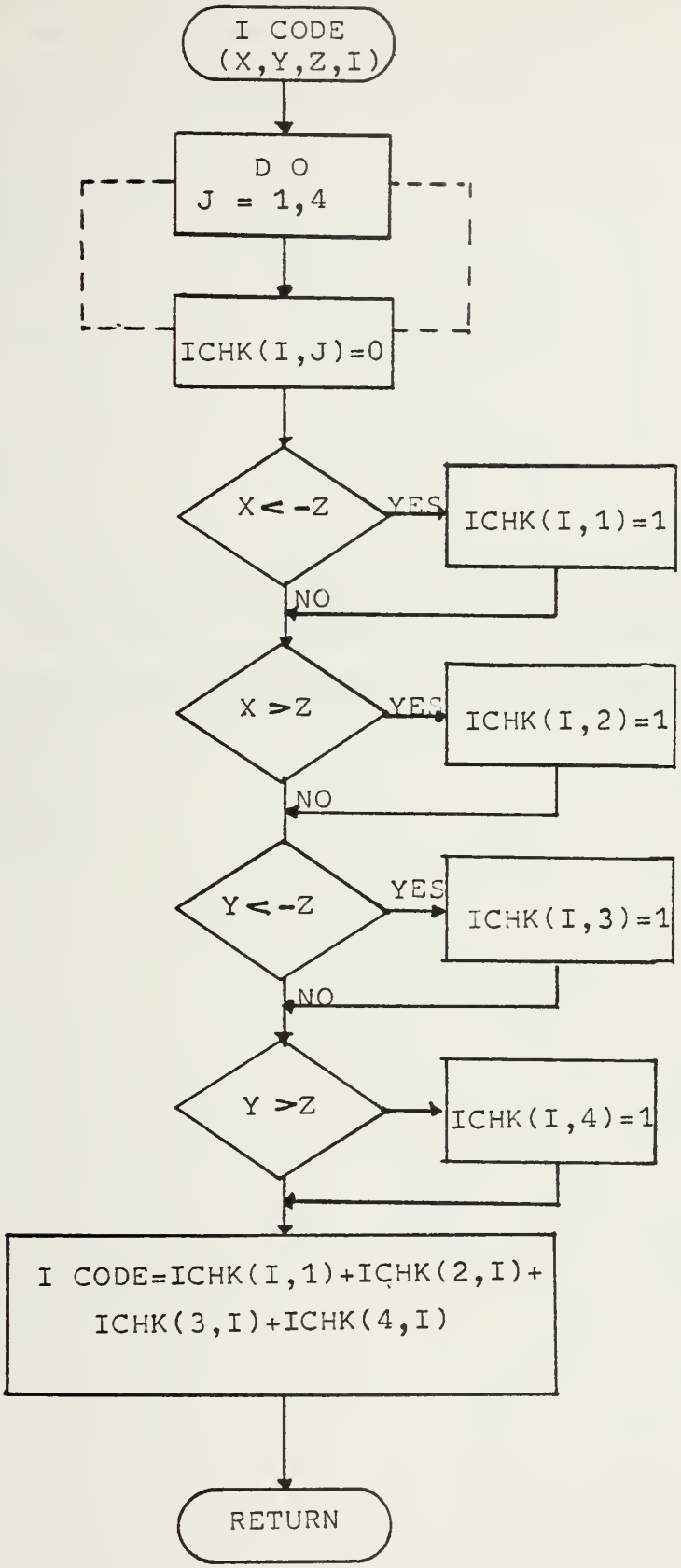

RDYCLP
ZERO ARRAY PCHANG
D O I=1,EDGEM
EDGE(1,I)=EDGE1(I)
EDGE(2,I)=EDGE2(I)
FORM MATRIX T1 (4,4)-TRANSLATES VIEWPOINT ORIGIN
FORM MATRIX T2 (4,4)-ESTABLISHES LEFT-HAND COORDINATE SYSTEM
FORM MATRIX T3 (4,4)-ROTATES VIEWING AXIS INTO THE X-Z PLANE
FORM MATRIX T4 (4,4)-ROTATES VIEWING AXIS INTO THE Z-AXIS
A
A
FORM VIEWING TRANSFORM: V=T1 T2 T3 T4
D O I=1,POINTM
[XS(I) YS(I) ZS(I) 1] = [XE(I) YE(I) ZE(I) 1] V
RETURN

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     CLIP: CLIPS THE IMAGE AGAINST THE VISIBLE DISPLAY
C        AND ELIMINATES ALL PORTIONS OF EDGES WHICH ARE OFF THE
C        SCREEN
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE CLIP
      DIMENSION J(2),X(2),Y(2),Z(2),ITOT(2)
      COMMON /AB/ POLYGN,POLGN,SHAD
      COMMON /AAD/EDGE,EDGEM
      COMMON /FF/ PCHANG,THETA
      COMMON /AAB/ XS(120),YS(120),ZS(120),POINTM
      COMMON /II/ ICHK(2,4)
      INTEGER POLYGN(60,11),EDGE(2,200),PCHANG(200),POLGN,EDGEM
     &,SHAD(60),POINTM
      DO 136 I=1,EDGEM
         IS=1
         LN=2
         J(1)=EDGE(1,I)
         J(2)=EDGE(2,I)
         J1=J(1)
         J2=J(2)
         X(1)=XS(J1)
         X(2)=XS(J2)
         Y(1)=YS(J1)
         Y(2)=YS(J2)
         Z(1)=ZS(J1)
         Z(2)=ZS(J2)
         ITOT(1)=ICODE(X(1),Y(1),Z(1),1)
         ITOT(2)=ICODE(X(2),Y(2),Z(2),2)
 148     IB=ITOT(1)+ITOT(2)
         IF(IB.EQ.0) GO TO 140
         DO 141 K=1,4
              IA=ICHK(1,K)+ICHK(2,K)
              IF(IA.EQ.2) GO TO 142
 141     CONTINUE
         IF(ITOT(IS).EQ.0) CALL ISWAP(IS,LN)
         IR=J(IS)
         PCHANG(IR)=1
         IF(ICHK(IS,1).EQ.0) GO TO 144
         T=(Z(IS)+X(IS))/((X(IS)-X(LN))-(Z(LN)-Z(IS)))
         Z(IS)=T*(Z(LN)-Z(IS))+Z(IS)
         X(IS)=-Z(IS)
         Y(IS)=T*(Y(LN)-Y(IS))+Y(IS)
         GO TO 147
 144     IF(ICHK(IS,2).EQ.0) GO TO 145
         T=(Z(IS)-X(IS))/((X(LN)-X(IS))-(Z(LN)-Z(IS)))
         Z(IS)=T*(Z(LN)-Z(IS))+Z(IS)
         X(IS)=Z(IS)
         Y(IS)=T*(Y(LN)-Y(IS))+Y(IS)
         GO TO 147
 145     IF(ICHK(IS,3).EQ.0) GO TO 146
         T=(Z(IS)+Y(IS))/((Y(IS)-Y(LN))-(Z(LN)-Z(IS)))
         Z(IS)=T*(Z(LN)-Z(IS))+Z(IS)
         Y(IS)=-Z(IS)
         X(IS)=T*(X(LN)-X(IS))+X(IS)
         GO TO 147
 146     T=(Z(IS)-Y(IS))/((Y(LN)-Y(IS))-(Z(LN)-Z(IS)))
         Z(IS)=T*(Z(LN)-Z(IS))+Z(IS)
         Y(IS)=Z(IS)
         X(IS)=T*(X(LN)-X(IS))+X(IS)
 147     ITOT(IS)=ICODE(X(IS),Y(IS),Z(IS),IS)
```

```
      GO TO 148
  140 IT=J(IS)
      IF(PCHANG(IT).EQ.0) GO TO 1488
      POINTM=POINTM+1
      IF(POINTM.GT.121) GO TO 149
      XS(POINTM)=X(IS)
      YS(POINTM)=Y(IS)
      ZS(POINTM)=Z(IS)
      EDGE(IS,I)=POINTM
 1488 IT=J(LN)
      IF(PCHANG(IT).EQ.0) GO TO 138
      POINTM=POINTM+1
      IF(POINTM.GT.120) GO TO 149
      XS(POINTM)=X(LN)
      YS(POINTM)=Y(LN)
      ZS(POINTM)=Z(LN)
      EDGE(LN,I)=POINTM
      GO TO 138
  142 EDGE(1,I)=0
      EDGE(2,I)=0
      IT=J(1)
      PCHANG(IT)=1
      IT=J(2)
      PCHANG(IT)=1
  138 CONTINUE
      RETURN
  149 WRITE(6,8)
    8 FORMAT(//1X,'THE NUMBER OF POINTS HAS EXCEEDED THE MAXIMUM
     1STORAGE ALLOCATED FOR POINTS'//)
      RETURN
      END
```

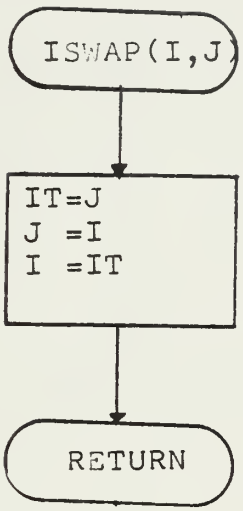

CLIP
I=1
IS=1
LN=2
C
I=I+1
GET TWO ENDPOINTS, J1 & J2, OF EDGE I. STORE XS,YS, & ZS VALUES IN ARRAYS X(2),Y(2)&Z(2)
COMPUTE LOCATION IN XS YS PLANE OF THE TWO ENDPOINTS
BOTH ENDPOINTS VIEWABLE
YES
A
NO
ENTIRE EDGE OFF SCREEN
YES
B
NO
ENDPOINT "IS" VIEWABLE
YES
ISWAP (IS,LN)
NO
PUSH ENDPOINT "IS" TOWARDS THE SCREEN
COMPUTE LOCATION OF NEW ENDPOINT "IS"
A
EXAMINE BOTH ENDPOINTS OF THE, NOW, VIEWABLE LINE SEGMENT.
EITHER ENDPOINT "PUSHED"
YES
E
NO
D
I=EDGEM
NO
C
YES
RETURN
B
DELETE ENTIRE EDGE
D
E
STORE X,Y & Z VALUES OF NEW ENDPOINT AT NEW XS(),YS(),&ZS() LOCATION
INCREMENT TOTAL NUMBER OF POINTS. STORE INDEX OF NEW POINT IN APPROPRIATE LOCATION FOR EDGE I.
D

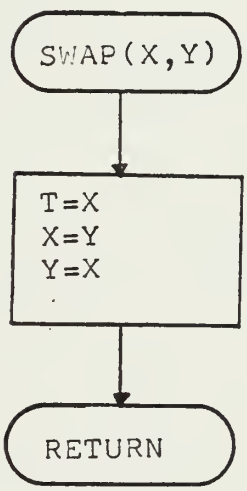
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     ICODE: USED BY CLIP TO DETERMINE IF AN END POINT IS VISIBLE AND
C        IF NOT TO WHICH SIDE OF THE DISPLAY THE POINT IS LOCATED.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      FUNCTION ICODE(X,Y,Z,I)
      COMMON /II/ICHK(2,4)
      DO 139 J=1,4
  139 ICHK(I,J)=0
      IF(X.LT.-Z) ICHK(I,1)=1
      IF(X.GT.Z) ICHK(I,2)=1
      IF(Y.LT.-Z) ICHK(I,3)=1
      IF(Y.GT.Z) ICHK(I,4)=1
      ICODE=ICHK(I,1)+ICHK(I,2)+ICHK(I,3)+ICHK(I,4)
      RETURN
      END
```


I CODE
(X,Y,Z,I)
D O
J = 1,4
ICHK(I,J)=0
X<-Z
YES
ICHK(I,1)=1
NO
X>Z
YES
ICHK(I,2)=1
NO
Y<-Z
YES
ICHK(I,3)=1
NO
Y>Z
ICHK(I,4)=1
I CODE=ICHK(I,1)+ICHK(2,I)+
ICHK(3,I)+ICHK(4,I)
RETURN

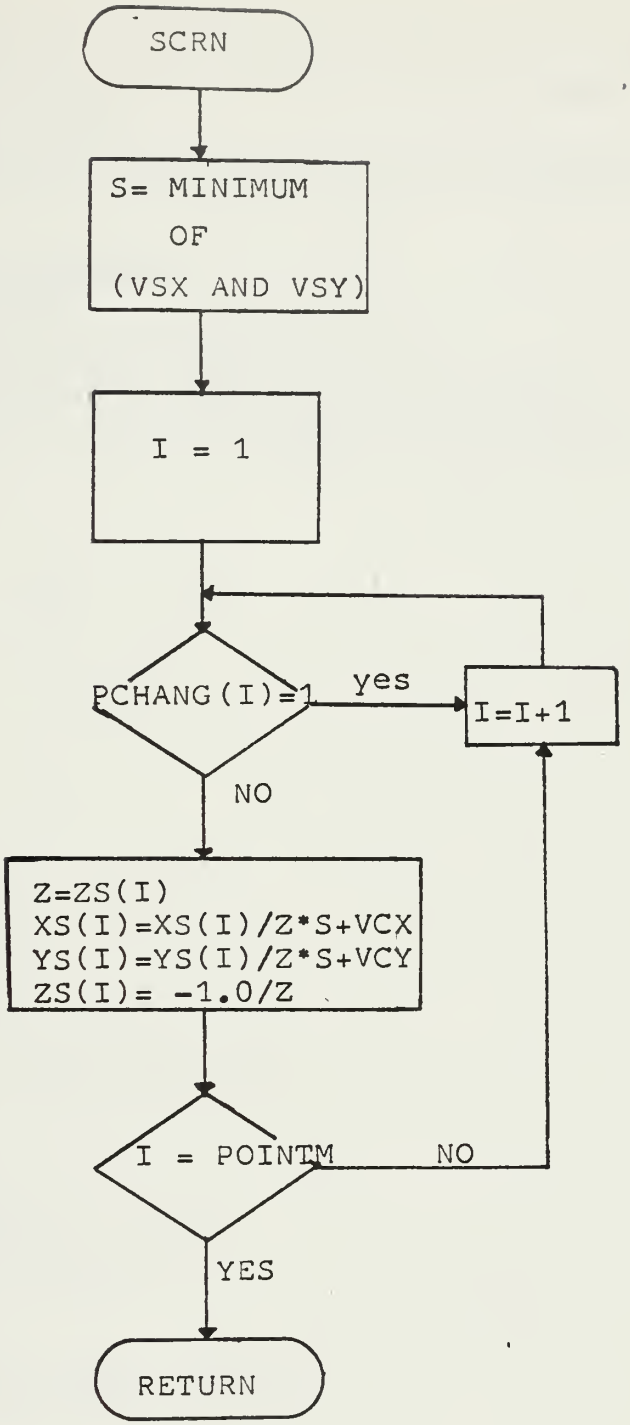
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       ISWAP: SWAPS THE VALUES OF TWO INTEGERS.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE ISWAP(I,J)
      IT=I
      I=J
      J=IT
      RETURN
      END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       SWAP: SWAPS THE VALUES OF TWO REAL NUMBERS.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE SWAP(X,Y)
      T=X
      X=Y
      Y=T
      RETURN
      END
```

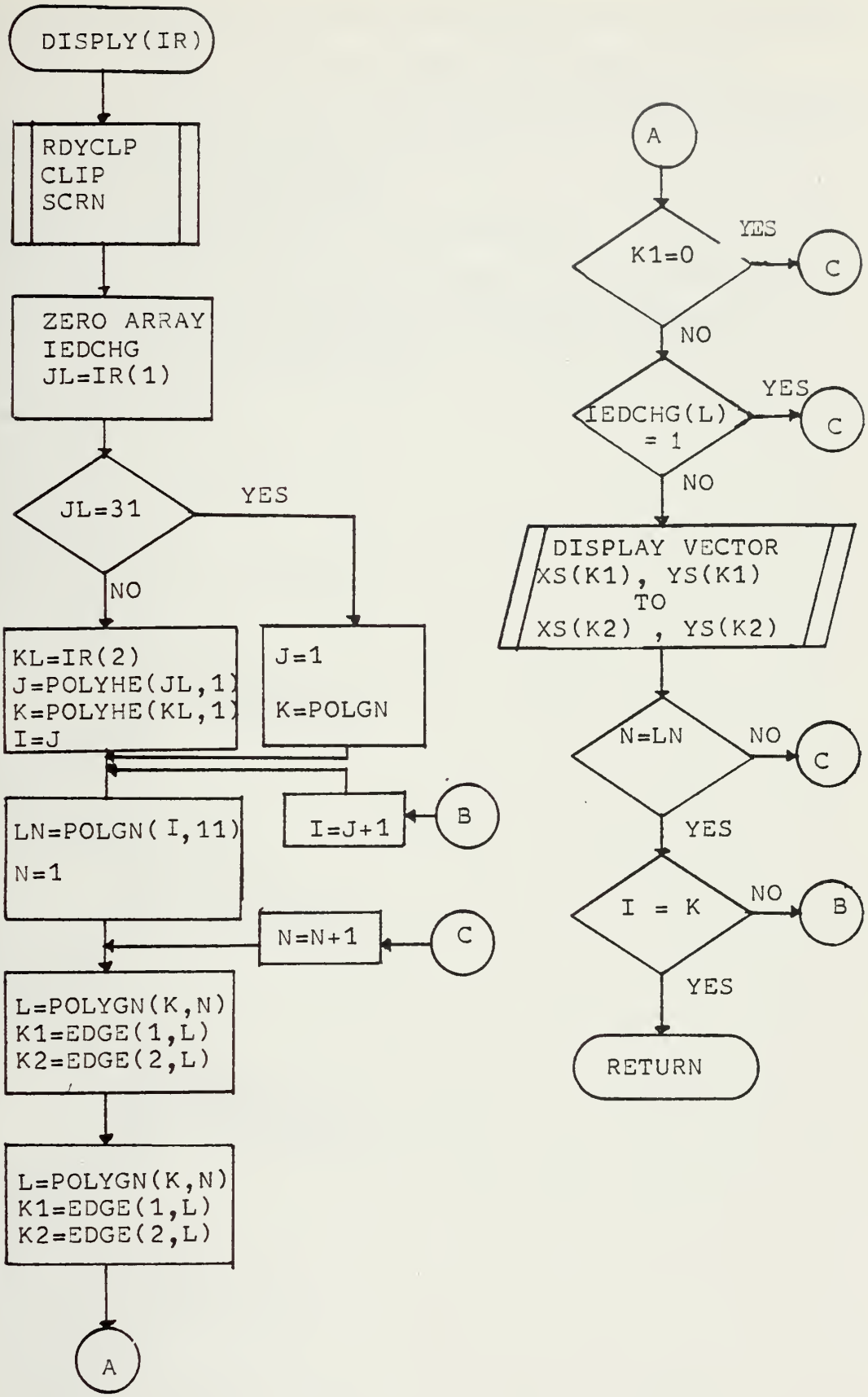

ISWAP(I,J)
IT=J
J =I
I =IT
RETURN


SWAP(X,Y)
T=X
X=Y
Y=X
RETURN

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      SCRN: TRANSFORMS EYE COURDINATES TO SCREEN COURDINATES
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
       SUBROUTINE SCRN
       COMMON /AAB/ XS(120),YS(120),ZS(120),POINTM
       COMMON /FF/ PCHANG
       COMMON /JJ/VSX,VSY,VCX,VCY
       INTEGER PCHANG(200),POINTM
       S=VSX
       IF(S.GT.VSY) S=VSY
       DO 151 I=1,POINTM
          IF(PCHANG(I).EQ.1) GO TO 151
          Z=ZS(I)
          XS(I)=XS(I)/Z*S+VCX
          YS(I)=YS(I)/Z*S+VCY
          ZS(I)=-1.0/Z
  151  CONTINUE
       RETURN
       END
```


SCRN
S= MINIMUM
OF
(VSX AND VSY)
I = 1
PCHANG(I)=1
yes
I=I+1
NO
Z=ZS(I)
XS(I)=XS(I)/Z*S+VCX
YS(I)=YS(I)/Z*S+VCY
ZS(I)= -1.0/Z
I = POINTM
NO
YES
RETURN

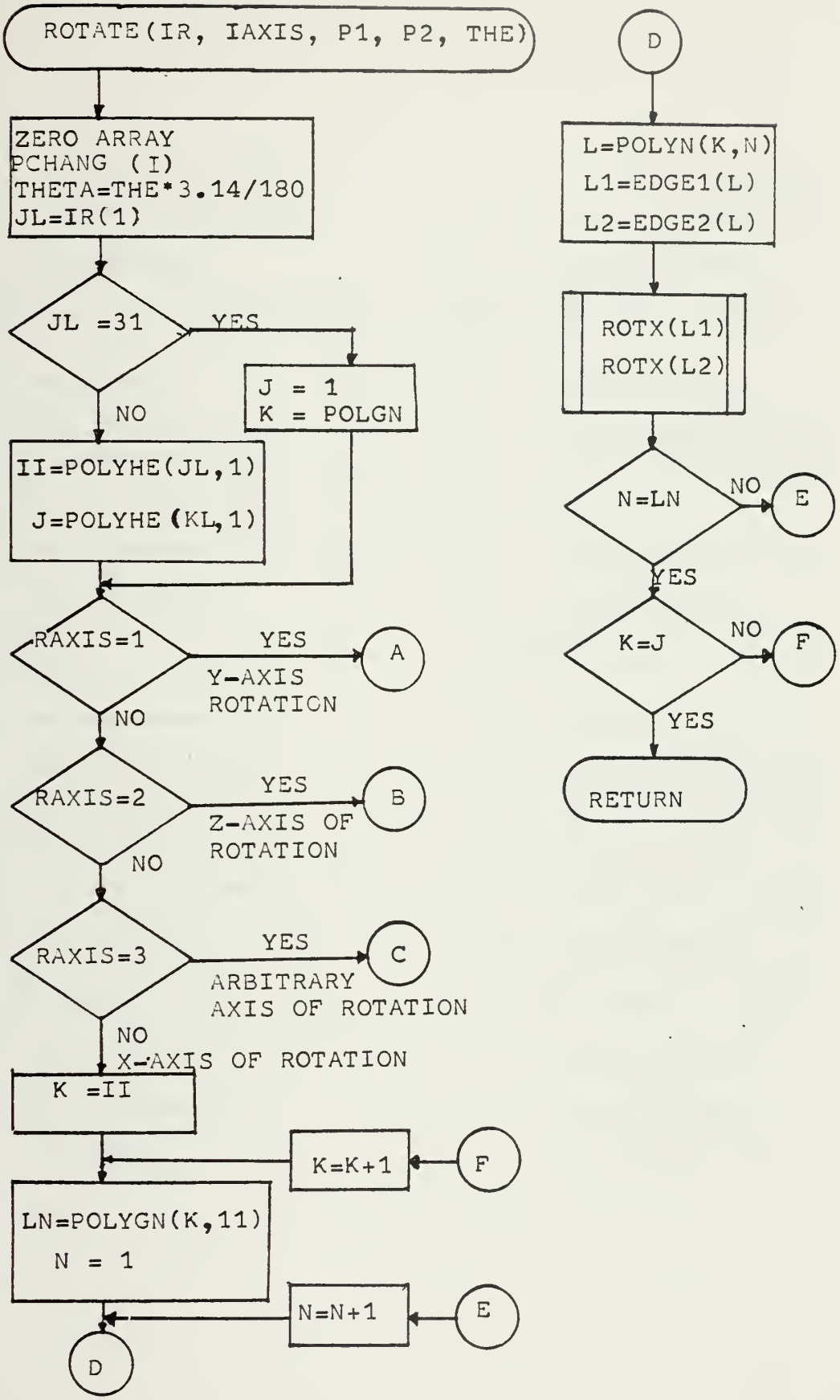
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     DISPLY: MASTER SUBROUTINE WHICH DRAWS THE CLIPPED IMAGE ON THE
C        SELECTED OUTPUT DEVICE.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE DISPLY(IR)
      DIMENSION IR(2)
      COMMON /AA/POLYHE,POLYHN
      COMMON /AB/POLYGN,POLGN,SHAD
      COMMON /AAD/ EDGE,EDGEM
      COMMON /AAB/ XS(120),YS(120),ZS(120),POINTM
      COMMON /FF/ EDGCHG(200)
      INTEGER POLYGN(60,11),EDGE(2,200),POLGN,EDGEM,COLOR
      INTEGER VECTOR,SHAD(60),POINTM,EDGCHG,POLYHE(10,2),POLYHN
      CALL RDYCLP
      CALL CLIP
      CALL SCRN
      DO 30 I=1,EDGEM
   30 EDGCHG(I)=0
      JL=IR(1)
      IF(JL.EQ.31) GO TO 152
      KL=IR(2)
      I=POLYHE(JL,1)
      J=POLYHE(KL,2)
      GO TO 1522
  152 I=1
      J=POLGN
 1522 JLM=SHAD(1)
      II=COLOR(JLM)
      DO 1530 K=I,J
        LN=POLYGN(K,11)
        DO 1540 N=1,LN
            L=POLYGN(K,N)
            K1=EDGE(1,L)
            K2=EDGE(2,L)
            IF(K1.EQ.0) GO TO 1540
            IF(EDGCHG(L).EQ.1) GO TO 1540
            EDGCHG(L)=1
            XS1=XS(K1)
            YS1=YS(K1)/2.0
            XS2=XS(K2)
            YS2=YS(K2)/2.0
            KR=VECTOR(XS1,YS1,XS2,YS2)
            IF(KR.LT.0) WRITE(6,3)
 1540     CONTINUE
 1530 CONTINUE
      RETURN
3     FORMAT('THE FUNCTION VECTOR FAILED')
      END
```

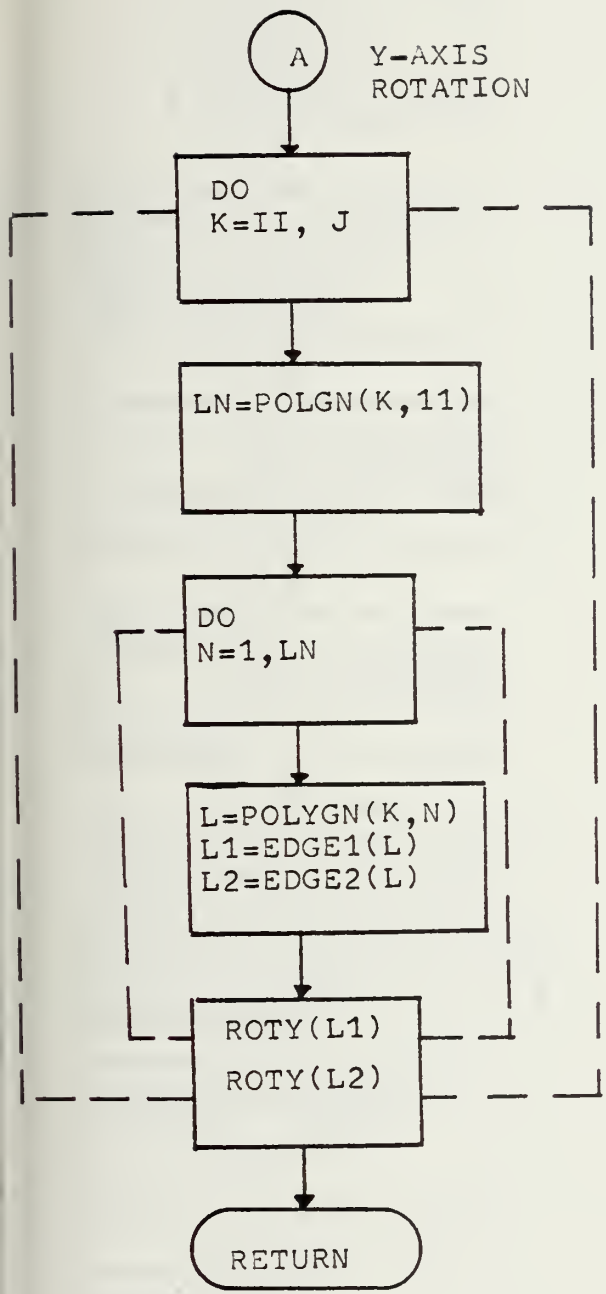

DISPLY(IR)
RDYCLP
CLIP
SCRN
ZERO ARRAY
IEDCHG
JL=IR(1)
JL=31
YES
NO
KL=IR(2)
J=POLYHE(JL,1)
K=POLYHE(KL,1)
I=J
J=1
K=POLGN
LN=POLGN(I,11)
N=1
I=J+1
B
N=N+1
C
L=POLYGN(K,N)
K1=EDGE(1,L)
K2=EDGE(2,L)
L=POLYGN(K,N)
K1=EDGE(1,L)
K2=EDGE(2,L)
A
A
K1=0
YES
C
NO
IEDCHG(L) = 1
YES
C
NO
DISPLAY VECTOR
XS(K1), YS(K1)
TO
XS(K2) , YS(K2)
N=LN
NO
C
YES
I = K
NO
B
YES
RETURN

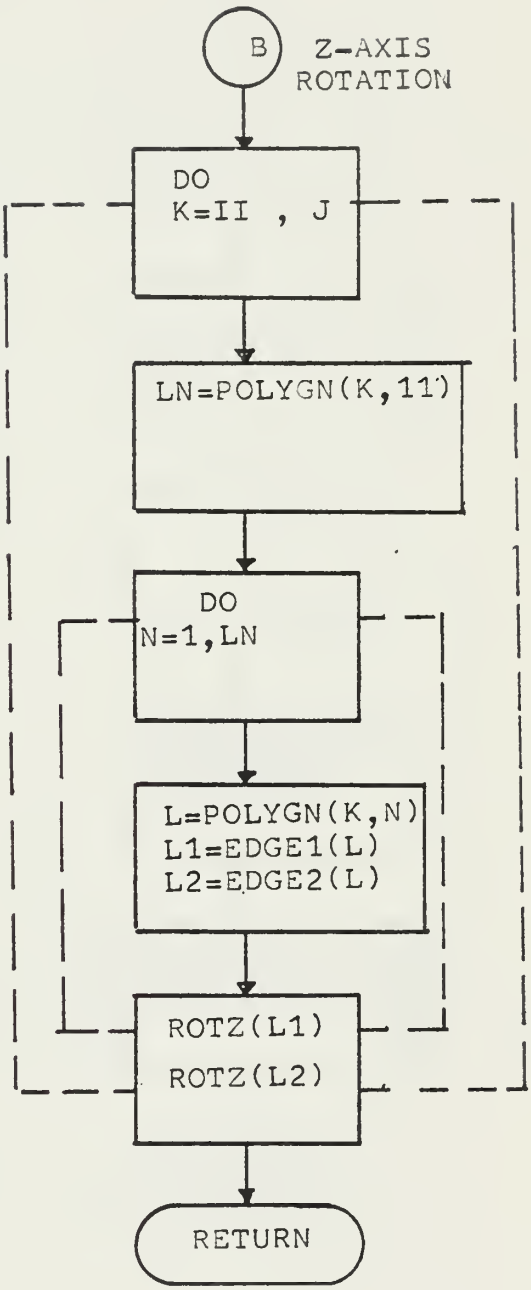
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     ROTATE: ROTATES A SINGLE POLYHEDRON OR THE ENTIRE IMAGE
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE ROTATE(IR,RAXIS,P1,P2,THE)
      COMMON /AA/ POLYHE, POLYHN
      COMMON /AB/ POLYGN,POLGN,SHAD
      COMMON /AC/ EDGE1,EDGE2,EDGEN
      COMMON /AAA/XE(120),YE(120),ZE(120),POINTN
      COMMON /FF/PCHANG,THETA
      DIMENSION T(4,4),R1(4,4),R2(4,4),R3(4,4),TT(4,4),TEMP(4,4),
     &LM(4),LR(4),P1(3),P2(3),IR(2)
      INTEGER POLYHE(10,2),POLYGN(60,11),EDGE1(100),EDGE2(100),
     &POLYHN,POLGN,EDGEN,POINTN,ROTIT,RAXIS,SHAD(60),PCHANG(200)
      DATA T,R1,R2,R3,TT,TEMP/96*0.0/
      DO 30 II=1,POINTN
   30 PCHANG(II)=0
      THETA=THE*3.14159/180.
      JL=IR(1)
      IF(JL.EQ.31)GO TO 111
      KL=IR(2)
      J=POLYHE(KL,2)
      II=POLYHE(JL,1)
      GO TO (112,113,114),RAXIS
      DO 115 K=II,J
         LN=POLYGN(K,11)
         DO 116 N=1,LN
              L=POLYGN(K,N)
              CALL ROTX(EDGE1(L))
              CALL ROTX(EDGE2(L))
  116    CONTINUE
  115 CONTINUE
      RETURN
  112 DO 118 K=II,J
         LN=POLYGN(K,11)
         DO 119 N=1,LN
              L=POLYGN(K,N)
              CALL ROTY(EDGE1(L))
              CALL ROTY(EDGE2(L))
  119    CONTINUE
  118 CONTINUE
      RETURN
  113 DO 120 K=II,J
         LN=POLYGN(K,11)
         DO 121 N=1,LN
              LN=POLYGN(K,N)
              CALL ROTZ(EDGE1(L))
              CALL ROTZ(EDGE2(L))
  121    CONTINUE
  120 CONTINUE
      RETURN
  114 DX=P2(1)-P1(1)
      DY=P2(2)-P1(2)
      DZ=P2(3)-P1(3)
      DTEMP=DX*DX+DY*DY+DZ*DZ
      DIST=SQRT(DTEMP)
      EX=DX/DIST
      EY=DY/DIST
      EZ=DZ/DIST
      V=SQRT(EY*EY+EZ*EZ)
      DO 122 L=1,4
         T(L,L)=1.0
```

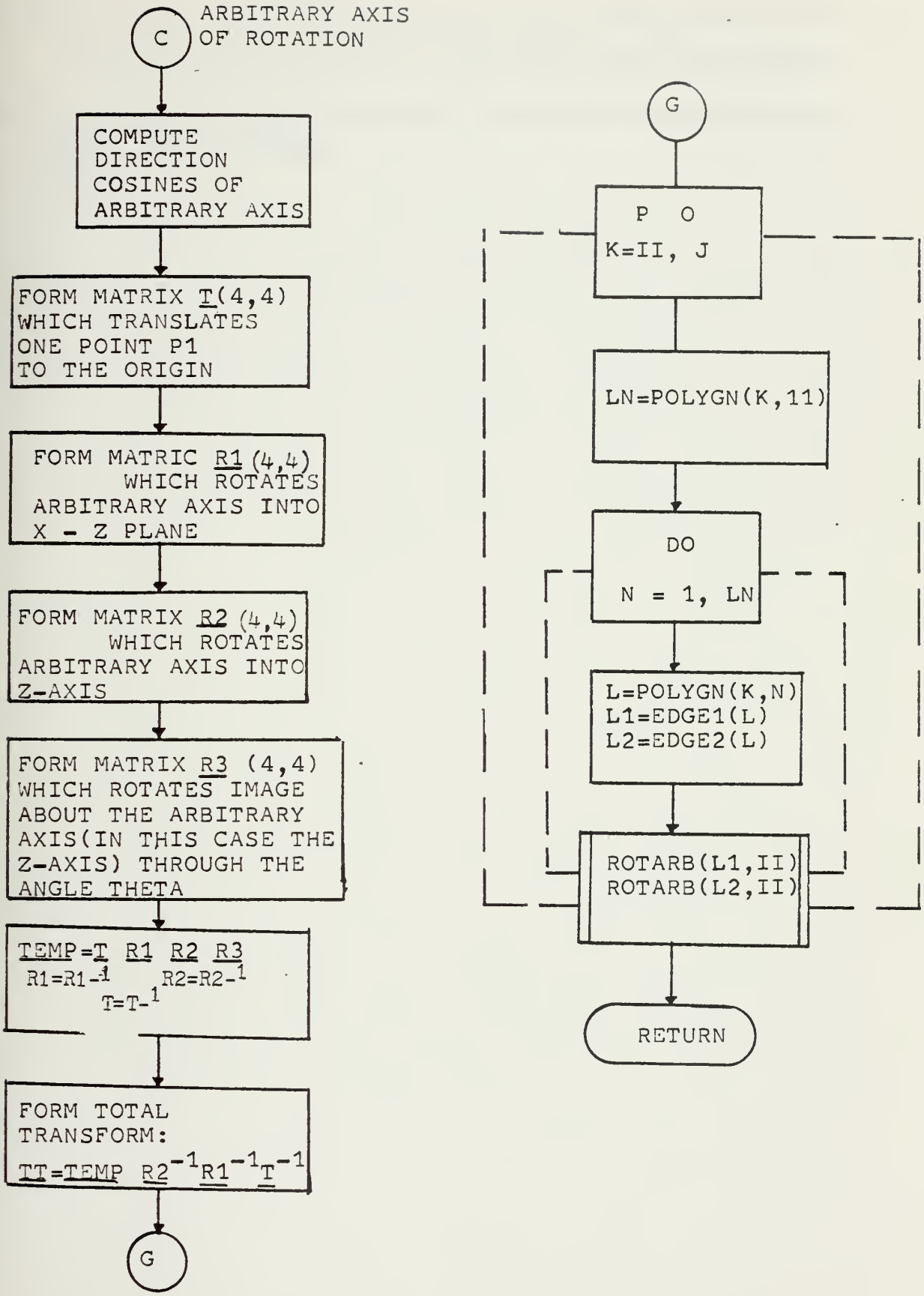
```
      R1(L,L)=1.0
      R2(L,L)=1.0
      R3(L,L)=1.0
122 CONTINUE
    AA=EZ/V
    BB=EY/V
    T(4,1)=-P1(1)
    T(4,2)=-P1(2)
    T(4,3)=-P1(3)
    R1(2,2)=AA
    R1(2,3)=BB
    R1(3,2)=-BB
    R1(3,3)=AA
    R2(1,1)=V
    R2(1,3)=EX
    R2(3,1)=-EX
    R2(3,3)=V
    R3(1,1)=COS(THETA)
    R3(1,2)=-SIN(THETA)
    R3(2,1)=-R3(1,2)
    R3(2,2)=R3(1,1)
    CALL GMPRD(T,R1,TEMP,4,4,4)
    CALL GMPRD(TEMP,R2,TT,4,4,4)
    CALL GMPRD(TT,R3,TEMP,4,4,4)
    RE=V*V+EX*EX
    RF=V/RE
    RG=EX/RE
    RD=AA*AA+BB*BB
    RA=AA/RD
    RB=BB/RD
    T(4,1)=P1(1)
    T(4,2)=P1(2)
    T(4,3)=P1(3)
    R1(2,2)=RA
    R1(2,3)=-RB
    R1(3,2)=RB
    R1(3,3)=RA
    R2(1,1)=RF
    R2(1,3)=-RG
    R2(3,1)=RG
    R2(3,3)=RF
    CALL GMPRD(TEMP,R2,TT,4,4,4)
    CALL GMPRD(TT,R1,TEMP,4,4,4)
    CALL GMPRD(TEMP,T,TT,4,4,4)
    IF(I.EQ.31) GO TO 123
    DO 124 K=1T,J
       LN=POLYGN(K,11)
       DO 125 N=1,LN
            L=POLYGN(K,N)
            CALL ROTARB(EDGE1(L),TT)
            CALL ROTARB(EDGE2(L),TT)
125    CONTINUE
124 CONTINUE
    RETURN
123 DO 126 K=1,POINTN
    CALL ROTARB(K,TT)
126 CONTINUE
    RETURN
111 GO TO (128,129,114),PAXIS
    DO 127 K=1,POINTN
       CALL ROTX(K)
127 CONTINUE
    RETURN
```

```
128 DO 130 K=1,POINTN
       CALL ROTY(K)
130 CONTINUE
    RETURN
129 DO 131 K=1,POINTN
       CALL ROTZ(K)
131 CONTINUE
    RETURN
    END
```

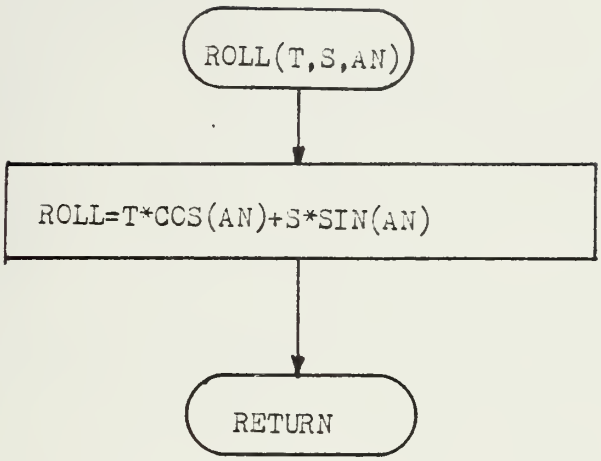

ROTATE(IR, IAXIS, P1, P2, THE)
ZERO ARRAY
PCHANG (I)
THETA=THE*3.14/180
JL=IR(1)
JL =31
YES
NO
J = 1
K = POLGN
II=POLYHE(JL,1)
J=POLYHE (KL,1)
RAXIS=1
YES
A
Y-AXIS
ROTATION
NO
RAXIS=2
YES
B
Z-AXIS OF
ROTATION
NO
RAXIS=3
YES
C
ARBITRARY
AXIS OF ROTATION
NO
X-AXIS OF ROTATION
K =II
K=K+1
F
LN=POLYGN(K,11)
N = 1
N=N+1
E
D
D
L=POLYN(K,N)
L1=EDGE1(L)
L2=EDGE2(L)
ROTX(L1)
ROTX(L2)
N=LN
NO
E
YES
K=J
NO
F
YES
RETURN


A
Y-AXIS
ROTATION
DO
K=II, J
LN=POLGN(K,11)
DO
N=1,LN
L=POLYGN(K,N)
L1=EDGE1(L)
L2=EDGE2(L)
ROTY(L1)
ROTY(L2)
RETURN

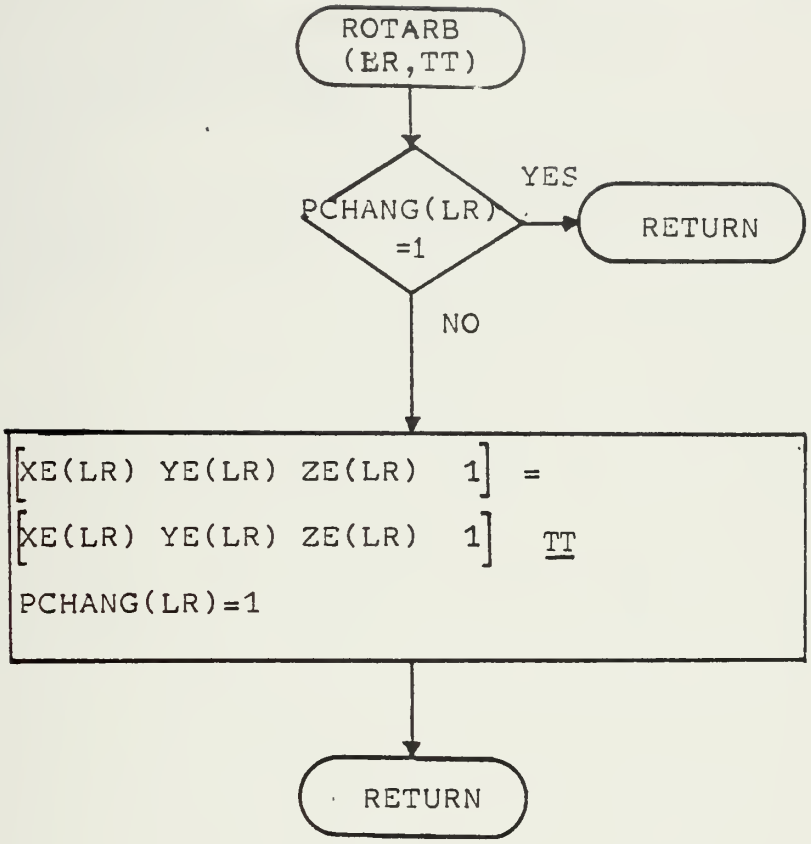

B
Z-AXIS
ROTATION
DO
K=II , J
LN=POLYGN(K,11)
DO
N=1,LN
L=POLYGN(K,N)
L1=EDGE1(L)
L2=EDGE2(L)
ROTZ(L1)
ROTZ(L2)
RETURN


C
ARBITRARY AXIS OF ROTATION
COMPUTE DIRECTION COSINES OF ARBITRARY AXIS
FORM MATRIX T(4,4) WHICH TRANSLATES ONE POINT P1 TO THE ORIGIN
FORM MATRIC R1 (4,4) WHICH ROTATES ARBITRARY AXIS INTO X - Z PLANE
FORM MATRIX R2 (4,4) WHICH ROTATES ARBITRARY AXIS INTO Z-AXIS
FORM MATRIX R3 (4,4) WHICH ROTATES IMAGE ABOUT THE ARBITRARY AXIS(IN THIS CASE THE Z-AXIS) THROUGH THE ANGLE THETA
TEMP=T R1 R2 R3
R1=R1-1 R2=R2-1
T=T-1
FORM TOTAL TRANSFORM:
TT=TEMP R2-1R1-1T-1
G
G
P O
K=II, J
LN=POLYGN(K,11)
DO
N = 1, LN
L=POLYGN(K,N)
L1=EDGE1(L)
L2=EDGE2(L)
ROTARB(L1,II)
ROTARB(L2,II)
RETURN

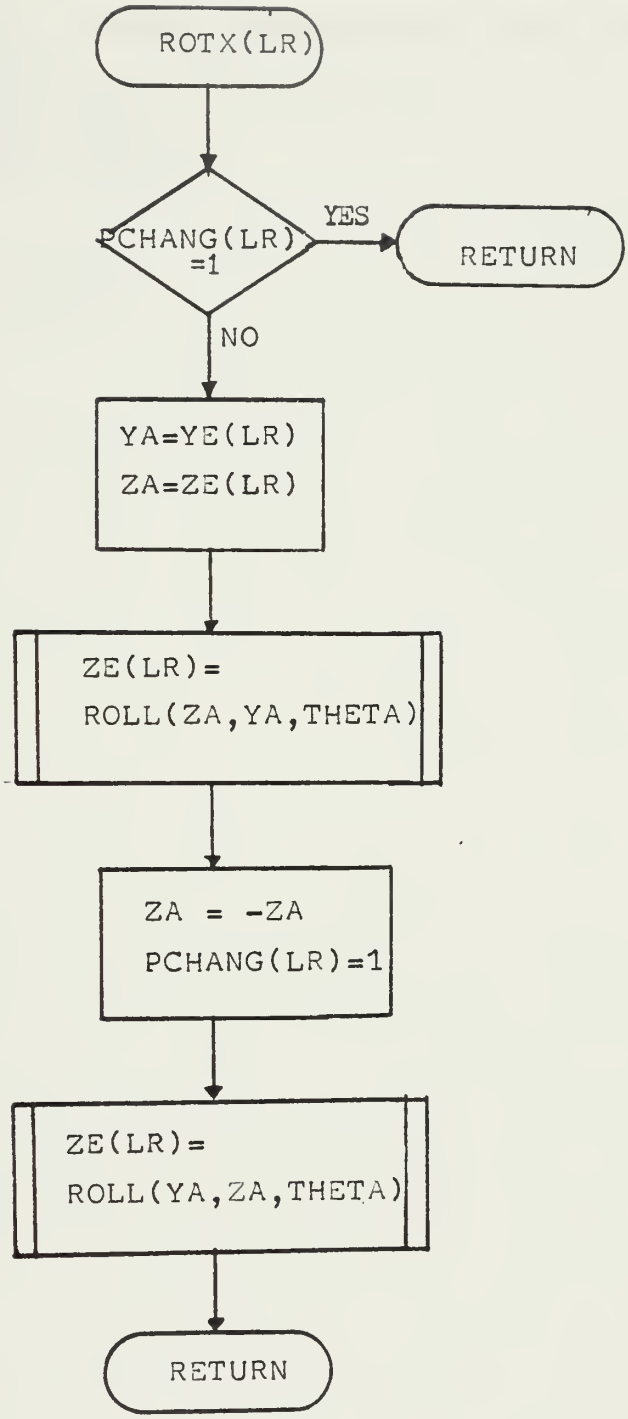
```
CCCC
```


ROLL(T,S,AN)
ROLL=T*COS(AN)+S*SIN(AN)
RETURN

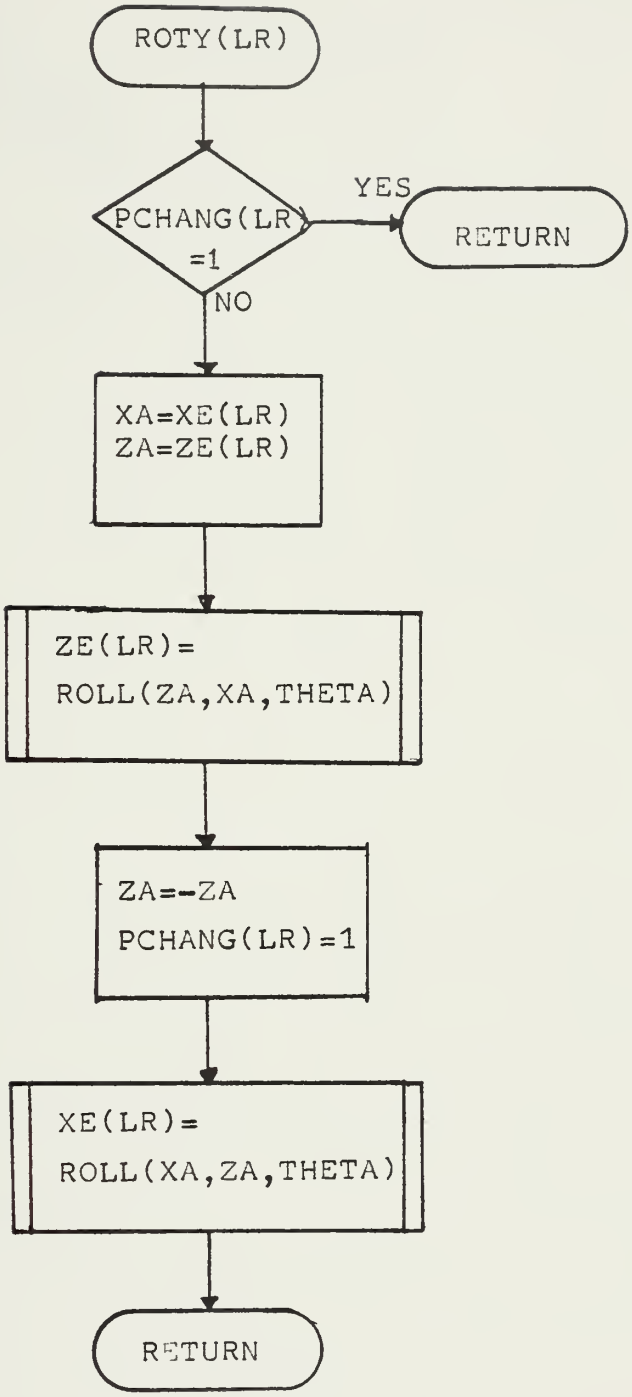
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     ROTARB: PERFORMS THE REPEATED MULT.'S TO ROTATE THE SELECTED
C        IMAGE ABOUT AN ARBITRARY AXIS
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE ROTARB(LR,TT)
      DIMENSION TT(4,4),TEMP(4),TNEW(4)
      COMMON /AAA/ XE(120),YE(120),ZE(120),POINTN
      COMMON /FF/ PCHANG,THETA
      INTEGER PCHANG(200),POINTN
      IF(PCHANG(LR).EQ.1) RETURN
      TEMP(1)=XE(LR)
      TEMP(2)=YE(LR)
      TEMP(3)=ZE(LR)
      TEMP(4)=1.0
      CALL GPRD(TEMP,TT,TNEW)
      XE(LR)=TNEW(1)
      YE(LR)=TNEW(2)
      ZE(LR)=TNEW(3)
      PCHANG(LR)=1
      RETURN
      END
```


ROTARB
(ER,TT)
PCHANG(LR)
=1
YES
RETURN
NO
[XE(LR) YE(LR) ZE(LR) 1] =
[XE(LR) YE(LR) ZE(LR) 1] TT
PCHANG(LR)=1
RETURN

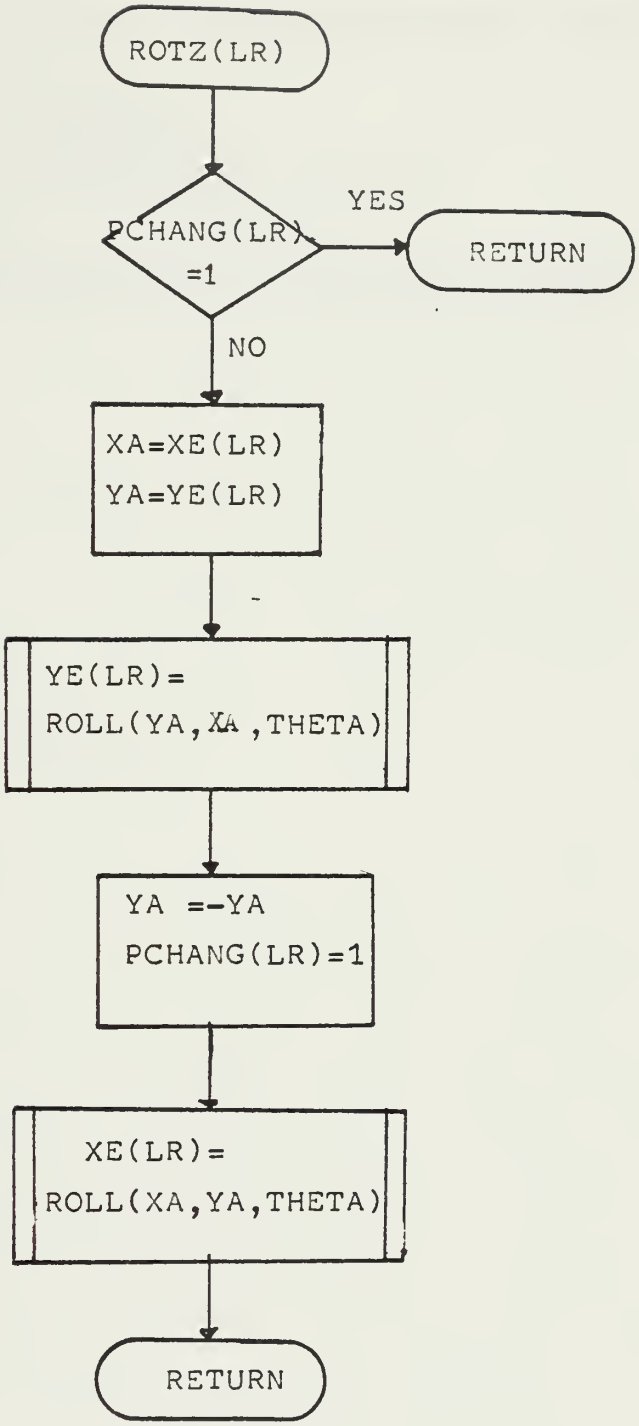
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     ROTX: PERFORMS THE REPEATED MULTIPLICATIONS TO ROTATE THE
C        SELECTED IMAGE ABOUT THE x AXIS
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE ROTx(LR)
      COMMON /AAA/ xE(120),YE(120),ZE(120),POINTN
      COMMON /FF/ PCHANG,THETA
      INTEGER PCHANG(200),POINTN
      IF(PCHANG(LR).EQ.1) RETURN
      YA=YE(LR)
      ZA=ZE(LR)
      ZE(LR)=ROLL(ZA,YA,THETA)
      ZA=-ZA
      YE(LR)=ROLL(YA,ZA,THETA)
      PCHANG(LR)=1
      RETURN
      END
```


ROTX(LR)
PCHANG(LR) =1
YES
RETURN
NO
YA=YE(LR)
ZA=ZE(LR)
ZE(LR)=
ROLL(ZA,YA,THETA)
ZA = -ZA
PCHANG(LR)=1
ZE(LR)=
ROLL(YA,ZA,THETA)
RETURN

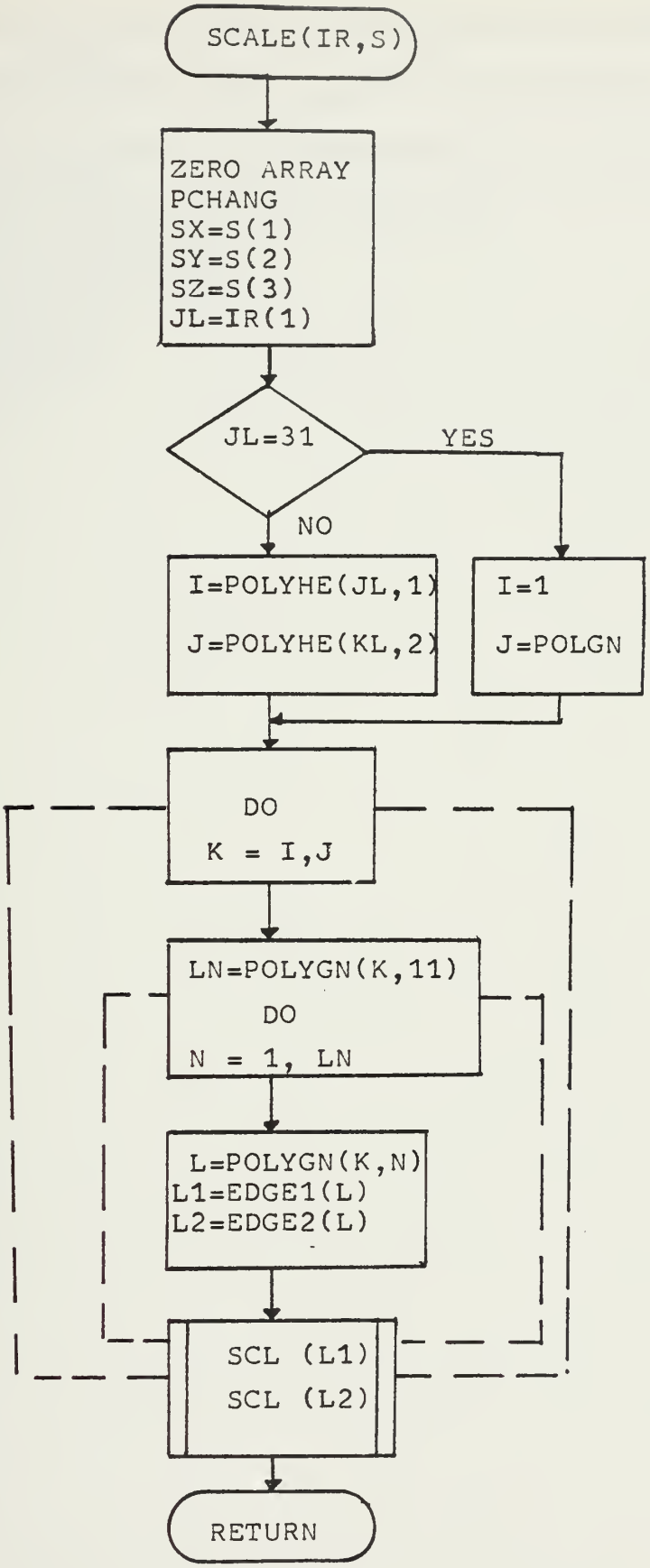
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     ROTY: PERFORMS THE REPEATED MULT.'S TO ROTATE THE SELECTED
C        IMAGE ABOUT THE Y AXIS
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE ROTY(LR)
      COMMON /AAA/ XE(120),YE(120),ZE(120),POINTN
      COMMON /FF/ PCHANG,THETA
      INTEGER PCHANG(200),POINTN
      IF(PCHANG(LR).EQ.1) RETURN
      XA=XE(LR)
      ZA=ZE(LR)
      ZE(LR)=ROLL(ZA,XA,THETA)
      ZA=-ZA
      XE(LR)=ROLL(XA,ZA,THETA)
      PCHANG(LR)=1
      RETURN
      END
```


ROTY(LR)
PCHANG(LR)
=1
YES
RETURN
NO
XA=XE(LR)
ZA=ZE(LR)
ZE(LR)=
ROLL(ZA,XA,THETA)
ZA=-ZA
PCHANG(LR)=1
XE(LR)=
ROLL(XA,ZA,THETA)
RETURN

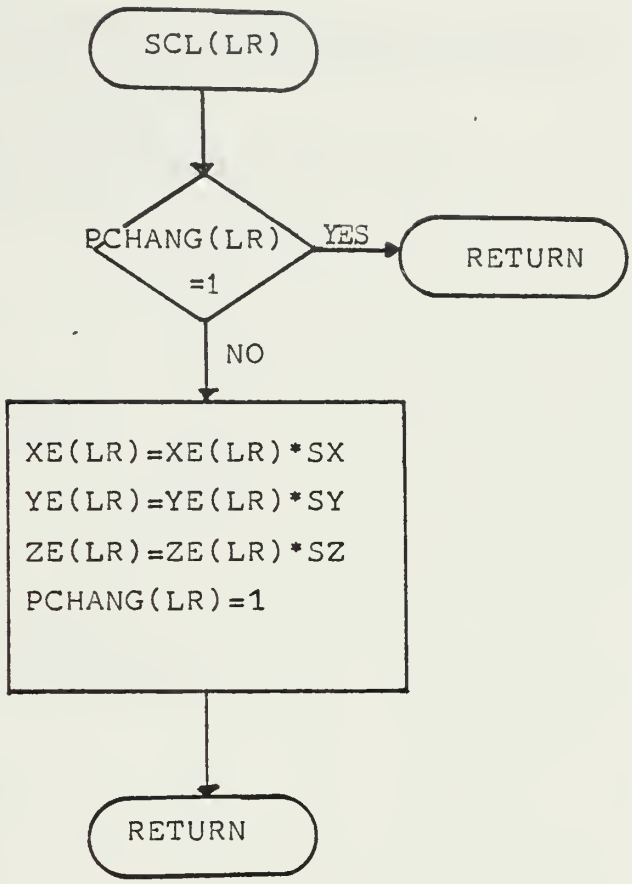
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     ROTZ: PERFORMS THE REPEATED MULT.'S TO ROTATE THE SELECTED
C        IMAGE ABOUT THE Z AXIS
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE ROTZ(LR)
      COMMON /AAA/ XE(120),YE(120),ZE(120),POINTN
      COMMON /FF/ PCHANG,THETA
      INTEGER PCHANG(200),POINTN
      IF(PCHANG(LR).EQ.1) RETURN
      XA=XE(LR)
      YA=YE(LR)
      YE(LR)=ROLL(YA,XA,THETA)
      YA=-YA
      XE(LR)=ROLL(XA,YA,THETA)
      PCHANG(LR)=1
      RETURN
      END
```


ROTZ(LR)
PCHANG(LR)
=1
YES
RETURN
NO
XA=XE(LR)
YA=YE(LR)
YE(LR)=
ROLL(YA,XA,THETA)
YA =-YA
PCHANG(LR)=1
XE(LR)=
ROLL(XA,YA,THETA)
RETURN

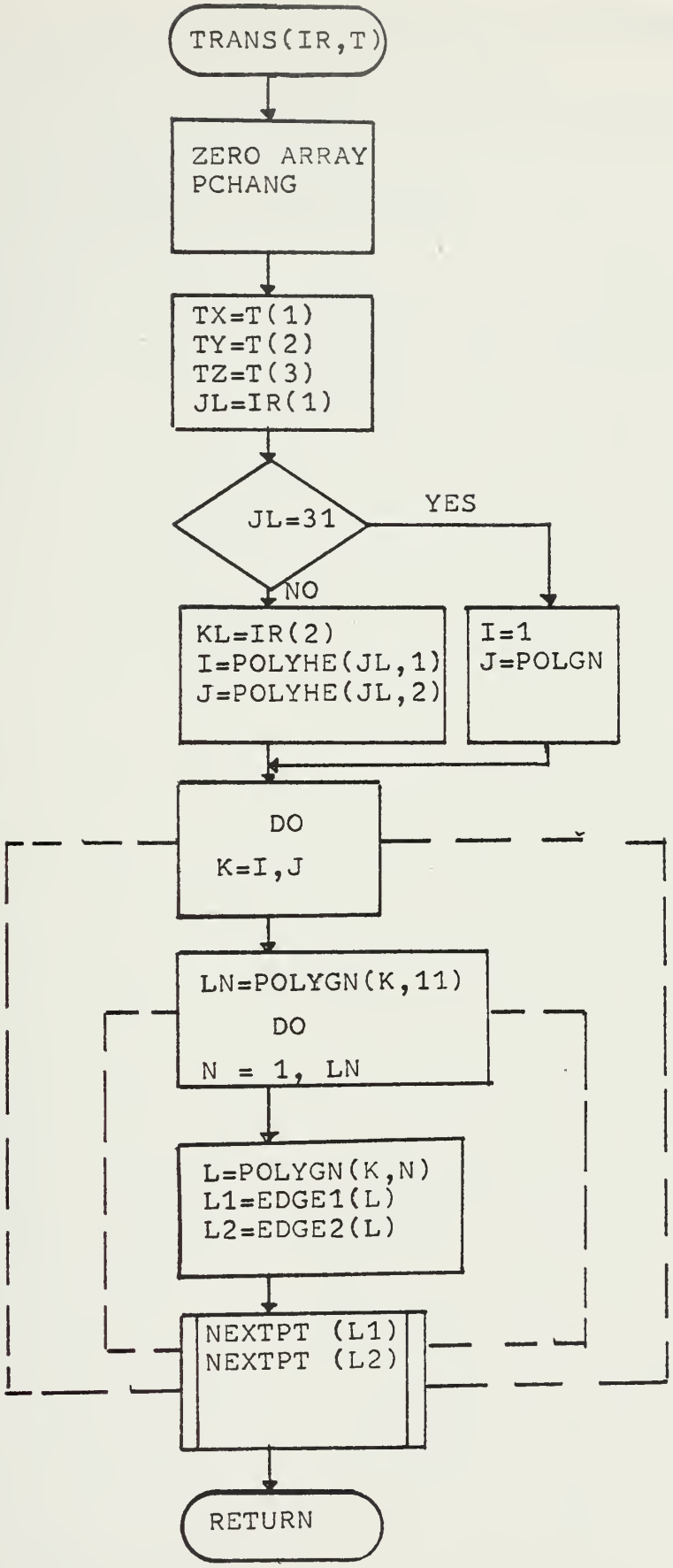
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     SCALE: SCALES A SINGLE POLYHEDRON OR THE ENTIRE IMAGE
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE SCALE(IR,S)
      DIMENSION S(3),IR(2)
      COMMON /AA/POLYHE,POLYHN
      COMMON /AB/ POLYGN,POLGN,SHAD
      COMMON /AC/ EDGE1,EDGE2,EDGEN
      COMMON /AAA/ XE(120),YE(120),ZE(120),POINTN
      COMMON /FF/PCHANG
      COMMON /HH/SX,SY,SZ
      INTEGER* POLYHE(10,2),POLYGN(60,11),EDGE1(100),EDGE2(100)
     &,POLYHN,POLGN,EDGEN,POINTN,SCALIT,SHAD(60),PCHANG(200)
      SX=S(1)
      SY=S(2)
      SZ=S(3)
      DO 30 II=1,POINTN
   30 PCHANG(II)=0
      JL=IR(1)
      IF(JL.EQ.31) GO TO 132
      KL=IR(2)
      J=POLYHE(KL,2)
      II=POLYHE(JL,1)
      DO 133 K=II,J
         LN=POLYGN(K,11)
         DO 134 N=1,LN
              L=POLYGN(K,N)
              CALL SCL(EDGE1(L))
              CALL SCL(EDGE2(L))
  134    CONTINUE
  133 CONTINUE
      RETURN
  132 DO 135 K=1,POINTN
         CALL SCL(K)
  135 CONTINUE
      RETURN
      END
```


SCALE(IR,S)
ZERO ARRAY
PCHANG
SX=S(1)
SY=S(2)
SZ=S(3)
JL=IR(1)
JL=31
YES
NO
I=POLYHE(JL,1)
J=POLYHE(KL,2)
I=1
J=POLGN
DO
K = I,J
LN=POLYGN(K,11)
DO
N = 1, LN
L=POLYGN(K,N)
L1=EDGE1(L)
L2=EDGE2(L)
SCL (L1)
SCL (L2)
RETURN

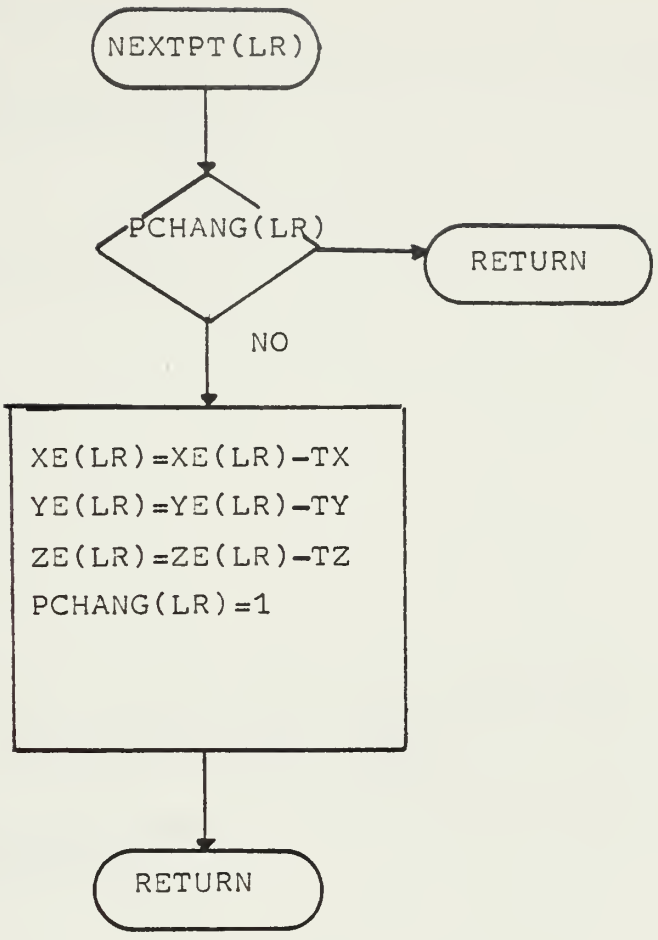
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      SCL: PERFORMS REPEATED MULT.'S TO SCALE THE SELECTED IMAGE
C         IN THE X, Y, AND Z DIRECTIONS
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
       SUBROUTINE SCL(LR)
       COMMON /AAA/ XE(120),YE(120),ZE(120),POINTN
       COMMON /HH/ SX,SY,SZ
       COMMON /FF/ PCHANG,THETA
       INTEGER PCHANG(200),POINTN
       IF(PCHANG(LR).EQ.1) RETURN
       XE(LR)=XE(LR)*SX
       YE(LR)=YE(LR)*SY
       ZE(LR)=ZE(LR)*SZ
       PCHANG(LR)=1
       RETURN
       END
```


SCL(LR)
PCHANG(LR)
=1
YES
RETURN
NO
XE(LR)=XE(LR)*SX
YE(LR)=YE(LR)*SY
ZE(LR)=ZE(LR)*SZ
PCHANG(LR)=1
RETURN

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     TRANSL: TRANSLATES A SINGLE POLYHEDRON OR THE ENTIRE IMAGE
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE TRANSL(IP,T)
      DIMENSION T(3),IR(2)
      INTEGER POLYHE(10,2),POLYGN(60,11),EDGE1(100),EDGE2(100),
     &PCHANG(200),POLGN,EDGEN,TRASIT,POINTN,POLYHN,SHAD(60)
      COMMON /AA/POLYHE,POLYHN
      COMMON /AB/POLYGN,POLGN,SHAD
      COMMON /AC/ EDGE1,EDGE2,EDGEN
      COMMON /AAA/ XE(120),YE(120),ZE(120),POINTN
      COMMON /FF/ PCHANG,THETA
      COMMON /EE/TX,TY,TZ
      TX=T(1)
      TY=T(2)
      TZ=T(3)
      DO 30 II=1,POINTN
   30 PCHANG(II)=0
      JL=IR(1)
      IF(JL.EQ.31) GO TO 103
      KL=IR(2)
      J=POLYHE(KL,2)
      II=POLYHE(JL,1)
      DO 105 K=I,J
         LN=POLYGN(K,11)
         DO 106 N=1,LN
              L=POLYGN(K,N)
              CALL NEXTPT(EDGE1(L))
              CALL NEXTPT(EDGE2(L))
  106    CONTINUE
  105 CONTINUE
      RETURN
  103 DO 107 K=1,POINTN
         CALL NEXTPT(K)
  107 CONTINUE
      RETURN
      END
```


TRANS(IR,T)
ZERO ARRAY
PCHANG
TX=T(1)
TY=T(2)
TZ=T(3)
JL=IR(1)
JL=31
YES
NO
KL=IR(2)
I=POLYHE(JL,1)
J=POLYHE(JL,2)
I=1
J=POLGN
DO
K=I,J
LN=POLYGN(K,11)
DO
N = 1, LN
L=POLYGN(K,N)
L1=EDGE1(L)
L2=EDGE2(L)
NEXTPT (L1)
NEXTPT (L2)
RETURN

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     NEXTPT: PERFORMS REPEATED OPERATIONS FOR TRANSL
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE NEXTPT(LR)
      INTEGER PCHANG(200),POINTN
      COMMON /AAA/XE(120),YE(120),ZE(120),POINTN
      COMMON /FF/PCHANG,THETA
      COMMON /EE/TX,TY,TZ
      IF(PCHANG(LR).EQ.1) RETURN
      XE(LR)=XE(LR)-TX
      YE(LR)=YE(LR)-TY
      ZE(LR)=ZE(LR)-TZ
      PCHANG(LR)=1
      RETURN
      END
```


NEXTPT(LR)
PCHANG(LR)
RETURN
NO
XE(LR)=XE(LR)-TX
YE(LR)=YE(LR)-TY
ZE(LR)=ZE(LR)-TZ
PCHANG(LR)=1
RETURN

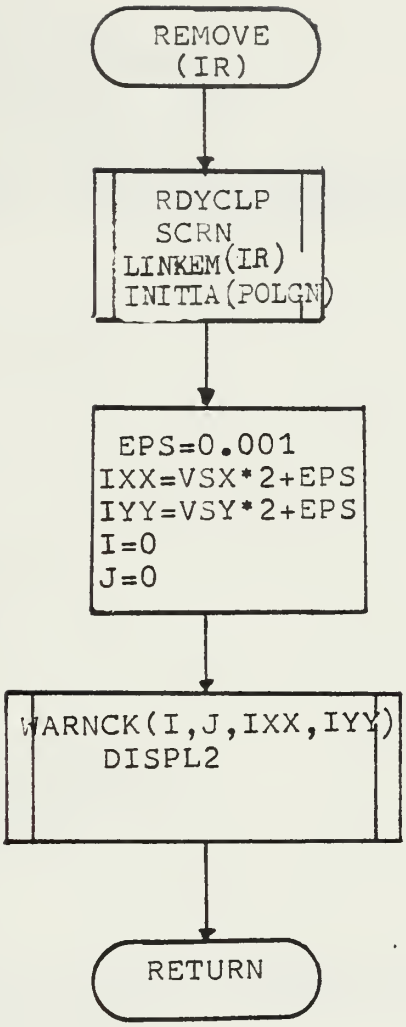
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       GMPRD: MULTIPLIES TWO 4 BY 4 MATRICES AND STORES THE RESULT
C           IN ANOTHER 4 BY 4 MATRIX.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE GMPRD(A,B,R,N,M,L)
      DIMENSION A(4,4),B(4,4),R(4,4)
      DO 10 K=1,4
      DO 10 J=1,4
      R(K,J)=0.0
      DO 10 I=1,4
10    R(K,J)=R(K,J)+A(K,I)*B(I,J)
      RETURN
      END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       GPRD: MULTIPLYS A 1 BY 4 VECTOR TIMES A 4 BY 4 MATRIX AND
C           STORES THE RESULT IN A 4 BY 1 VECTOR.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE GPRD(A,B,R)
      DIMENSION A(4),B(4,4),R(4)
      DO 10 I=1,4
      R(I)=0.0
      DO 10 J=1,4
10    R(I)=R(I)+A(J)*B(J,I)
      RETURN
      END
```

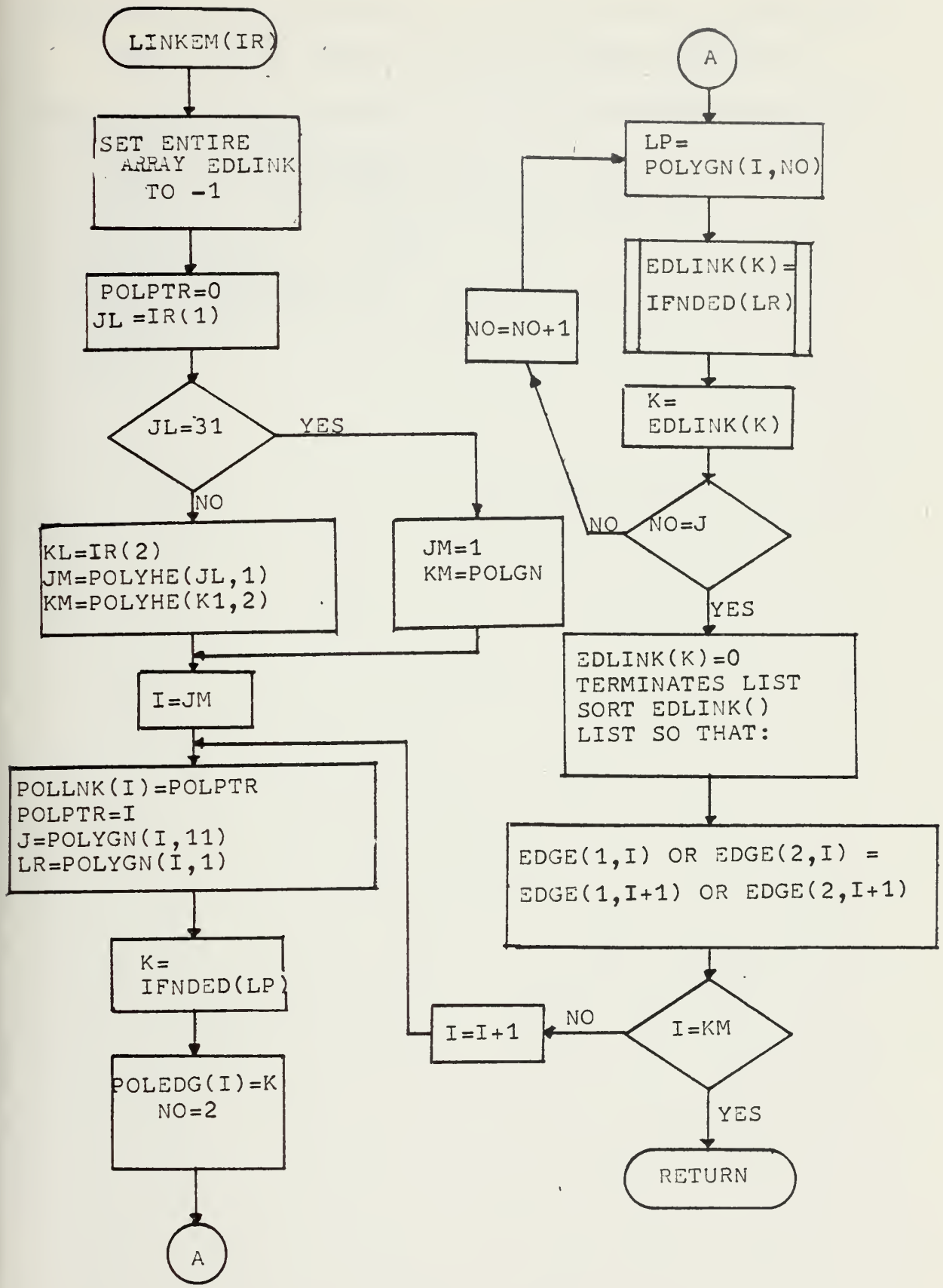
2. Hidden Line Removal

Included in this group were the subroutines utlized to remove hidden lines from the wire-frame figure produced by DISPLY. The subroutine REMOVE initiated this procedure when called.

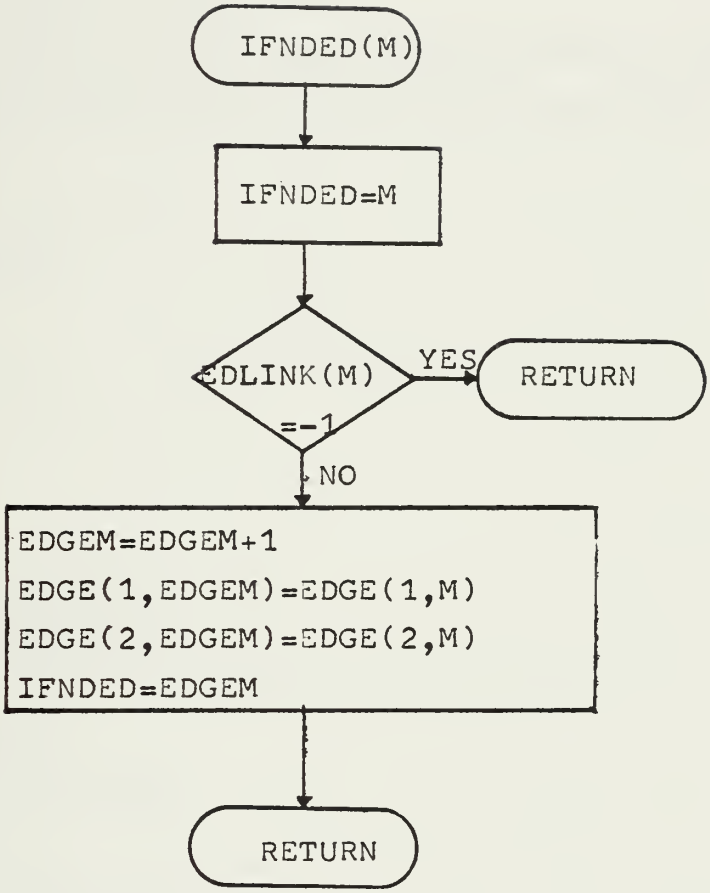
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      REMOVE: MASTER SUBROUTINE WHICH DRAWS THE IMAGE, AFTER REMOVING
C         ALL HIDDEN LINES, ON THE SELECTED OUTPUT DEVICE.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE REMOVE(IR)
      COMMON /AA/POLYHE(10,2),POLYHN
      DIMENSION IR(2)
      COMMON /AB/POLYGN,POLGN,SHAD
      COMMON /JJ/ VSX,VSY,VCX,VCY
      COMMON /DB/ JD
      INTEGER POLYGN(60,11),POLGN,SHAD(60),POLYHE,POLYHN
      CALL RDYCLP
      CALL SCRN
      CALL LINKEM(IR)
      CALL INITIA(POLGN)
      XX=VSX*2.+0.0001
      IXX=XX
      YY=VSY*2.+0.0001
      IYY=YY
      IPP=0
      JPP=0
      CALL WARNCK(IPP,JPP,IXX,IYY)
      JL=IR(1)
      IP=POLYHE(JL,1)
      ISHAD=SHAD(IP)
      CALL DISPL2(ISHAD)
      WRITE(6,1) JD
    1 FORMAT(1X,'THE NUMBER OF STORAGE LOCATIONS NEEDED IS=',I6)
      RETURN
      END
```


REMOVE
(IR)
RDYCLP
SCRN
LINKEM(IR)
INITIA(POLGN)
EPS=0.001
IXX=VSX*2+EPS
IYY=VSY*2+EPS
I=0
J=0
WARNCK(I,J,IXX,IYY)
DISPL2
RETURN

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     LINKEM: GENERATES LINKED LISTS FOR THE POLYGONS AND EDGES
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE LINKEM(IR)
      DIMENSION IR(2)
      COMMON /AA/POLYHE,POLYHN
      COMMON /AB/ POLYGN,POLGN,SHAD
      COMMON /AAD/ EDGE,EDGEM
      COMMON /RA/ EDLINK,POLEDG,NEXTED
      COMMON /RB/ POLLNK,POLPTR
      INTEGER POLYGN(60,11),EDGE(2,200),EDLINK(200),POLEDG(60)
     &,POLGN,EDGEM,POLPTR,SHAD(60),POLYHE(10,2),POLYHN,POLLNK(60)
      DO 31 I=1,200
   31 EDLINK(I)=-1
      POLPTR=0
      JL=IR(1)
      IF(JL.EQ.31) GO TO 2000
      KL=IR(2)
      JM=POLYHE(JL,1)
      KM=POLYHE(KL,2)
      GO TO 2020
 2000 JM=1
      KM=POLGN
 2020 DO 200 I=JM,KM
         POLLNK(I)=POLPTR
         POLPTR=I
         J=POLYGN(I,11)
         LR=POLYGN(I,1)
         K=IFNDED(LR)
         POLEDG(I)=K
         DO 201 NO=2,J
              LR=POLYGN(I,NO)
              EDLINK(K)=IFNDED(LR)
              K=EDLINK(K)
  201    CONTINUE
         EDLINK(K)=0
         K=POLEDG(I)
         J=EDGE(2,K)
         JT=EDLINK(K)
         IF(J.NE.EDGE(1,JT).AND.J.NE.EDGE(2,JT)) CALL ISWAP
     &   (EDGE(1,K),EDGE(2,K))
  202    IF(K.EQ.0) GO TO 200
         J=EDLINK(K)
         IF(J.NE.0.AND.EDGE(2,K).NE.EDGE(1,J)) CALL
     &   ISWAP(EDGE(1,J),EDGE(2,J))
         K=J
         GO TO 202
  200 CONTINUE
      RETURN
      END
```

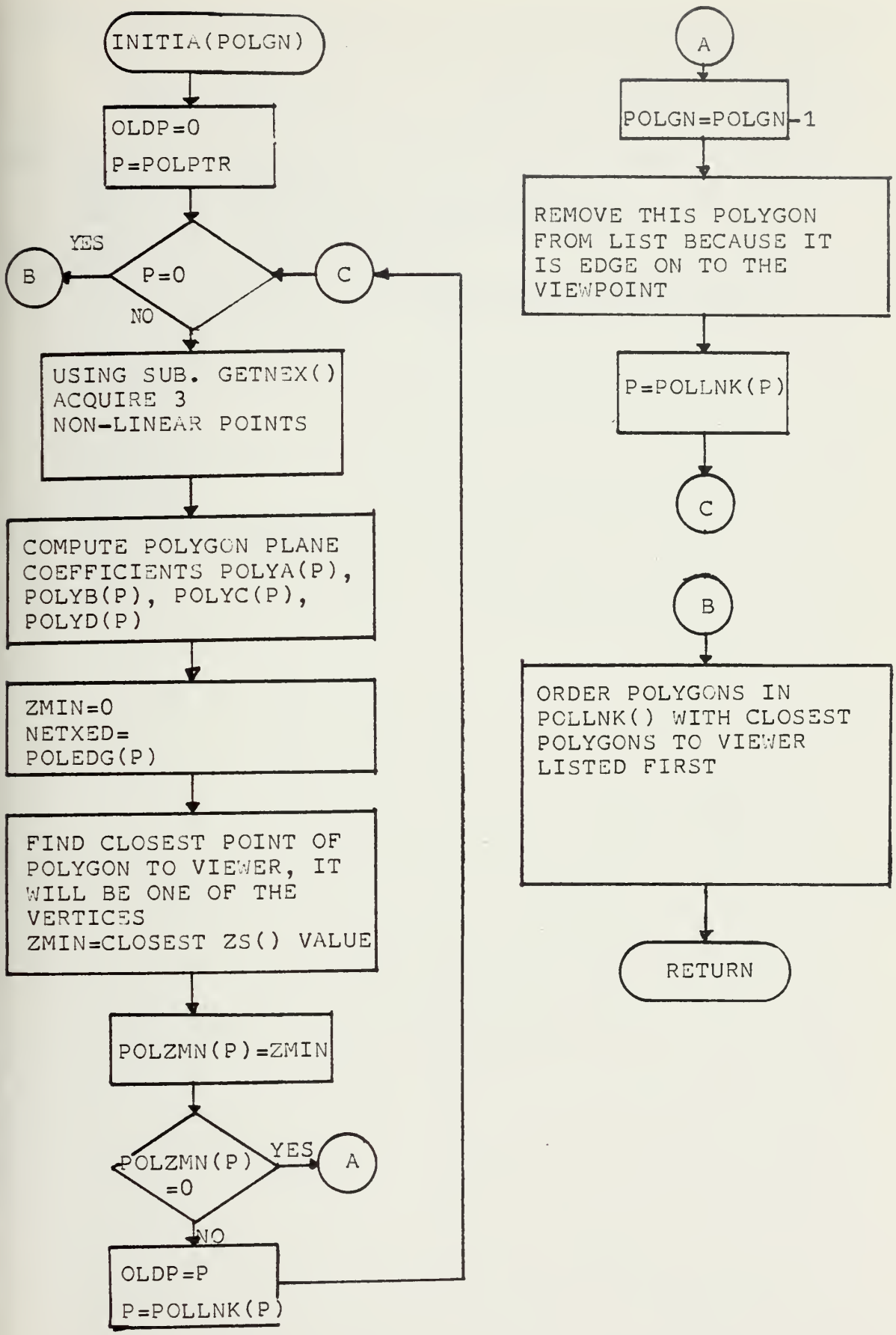

LINKEM(IR)
SET ENTIRE ARRAY EDLINK TO -1
POLPTR=0
JL =IR(1)
JL=31
YES
NO
KL=IR(2)
JM=POLYHE(JL,1)
KM=POLYHE(K1,2)
JM=1
KM=POLGN
I=JM
POLLNK(I)=POLPTR
POLPTR=I
J=POLYGN(I,11)
LR=POLYGN(I,1)
K=
IFNDED(LP)
POLEDG(I)=K
NO=2
A
I=I+1
NO
I=KM
YES
RETURN
A
LP=
POLYGN(I,NO)
EDLINK(K)=
IFNDED(LR)
K=
EDLINK(K)
NO
NO=J
YES
NO=NO+1
EDLINK(K)=0
TERMINATES LIST
SORT EDLINK()
LIST SO THAT:
EDGE(1,I) OR EDGE(2,I) =
EDGE(1,I+1) OR EDGE(2,I+1)

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      IFNDED: USED BY LINKEM TO FIND EMPTY STORAGE LOCATIONS IN
C         THE LIST USED TO LINK THE EDGES OF EACH POLYGON.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      FUNCTION IFNDED(M)
      COMMON /AAD/ EDGE,EDGEM
      COMMON /BA/ EDLINK,POLEDG,NEXTED
      INTEGER EDGE(2,200),EDGEM,EDLINK(200),POLEDG(60)
      IFNDED=M
      IF(EDLINK(M).EQ.-1) RETURN
      EDGEM=EDGEM+1
      EDGE(1,EDGEM)=EDGE(1,M)
      EDGE(2,EDGEM)=EDGE(2,M)
      IFNDED=EDGEM
      RETURN
      END
```

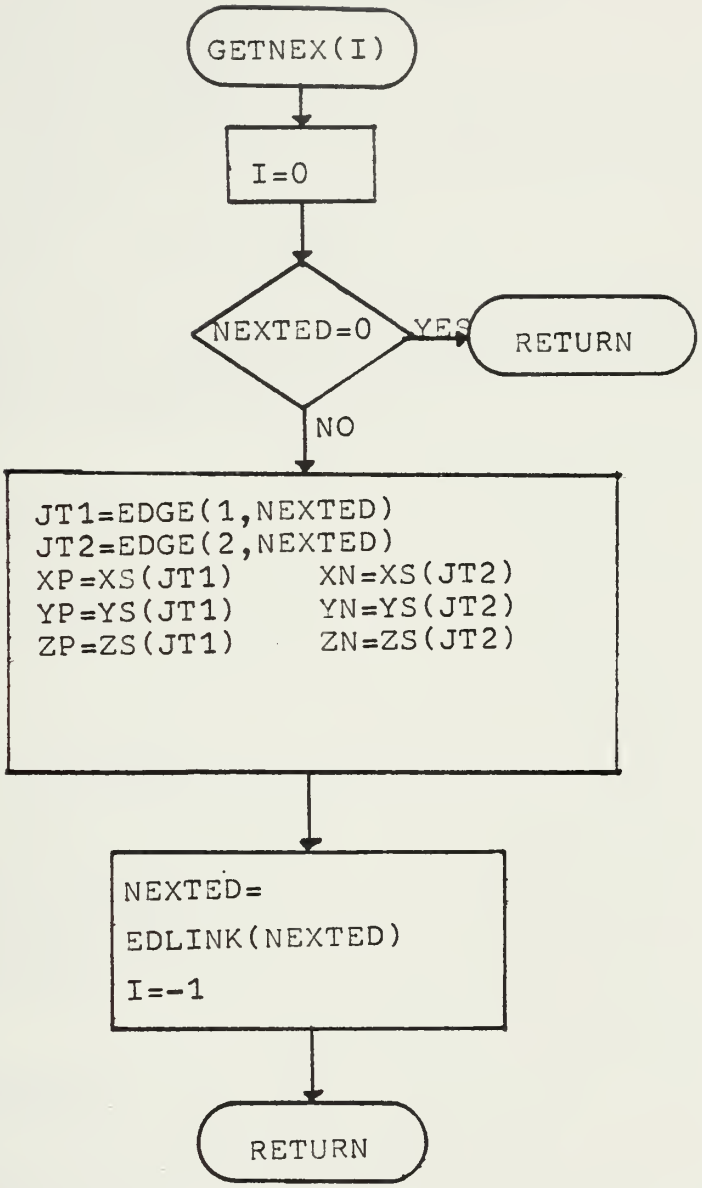

IFNDED(M)
IFNDED=M
EDLINK(M)
=-1
YES
RETURN
NO
EDGEM=EDGEM+1
EDGE(1,EDGEM)=EDGE(1,M)
EDGE(2,EDGEM)=EDGE(2,M)
IFNDED=EDGEM
RETURN

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     INITIA: SORTS THE LISTS OF EDGES FOR EACH POLYGON SO THAT THE END
C         POINT OF ONE EDGE IS COINCEDENT WITH THE FIRST POINT OF THE
C         NEXT EDGE ON THE LIST.  ADDITIONALLY, EACH POLYGON IS SORTED
C         WITH THE POLYGON CLOSEST TO THE VIEWER FIRST ON THE LIST AND
C         THE NEXT CLOSEST SECOND, ETC.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE INITIA(POLGN)
      COMMON /RA/ EDLINK,POLEDG,NEXTED
      COMMON /RB/ POLLNK,POLPTR
      COMMON /RC/ POLYA(60),POLYB(60),POLYC(60),POLYD(60),POLZMN(60)
      COMMON /RD/ XP,YP,ZP,XN,YN,ZN
      INTEGER EDLINK(200),POLEDG(60),POLLNK(60),POLPTR,OLDP
      INTEGER P,POLGN,CHANGE
      OLDP=0
      P=POLPTR
  207 IF(P.EQ.0) GO TO 203
      NEXTED=POLEDG(P)
      CALL GETNEX(I)
      X1=XP
      Y1=YP
      Z1=ZP
      CALL GETNEX(I)
      X3=XN-X1
      Y3=YN-Y1
      Z3=ZN-Z1
      X2=XP-X1
      Y2=YP-Y1
      Z2=ZP-Z1
      POLYA(P)=Y3*Z2-Y2*Z3
      POLYB(P)=X2*Z3-X3*Z2
      POLYC(P)=X3*Y2-X2*Y3
      POLYD(P)=-(POLYA(P)*X1+POLYB(P)*Y1+POLYC(P)*Z1)
      ZMIN=0.
      NEXTED=POLEDG(P)
  205 CALL GETNEX(I)
      IF(I.EQ.0) GO TO 204
      IF(ZMIN.GT.ZP)ZMIN=ZP
      GO TO 205
  204 POLZMN(P)=ZMIN
      IF(POLYC(P).EQ.0.)GO TO 206
      OLDP=P
  208 P=POLLNK(P)
      GO TO 207
  206 POLGN=POLGN-1
      IF(OLDP.EQ.0) GO TO 209
      POLLNK(OLDP)=POLLNK(P)
      GO TO 208
  209 POLPTR=POLLNK(P)
      GO TO 208
  203 CHANGE=0
      OLDP=0
      P=POLPTR
  215 IF(P.EQ.0) GO TO 211
      J=POLLNK(P)
      IF(J.EQ.0.OR.POLZMN(P).LE.POLZMN(J)) GO TO 212
      IF(OLDP.EQ.0) GO TO  213
      POLLNK(OLDP)=J
  214 POLLNK(P)=POLLNK(J)
      POLLNK(J)=P
      CHANGE=1
```

```
      OLDP=J
      GO TO 215
  213 POLPTR=J
      GO TO 214
  212 OLDP=P
      P=POLLNK(P)
      GO TO 215
  211 IF(CHANGE.EQ.1) GO TO 203
      RETURN
      END
```


INITIA(POLGN)
OLDP=0
P=POLPTR
YES
B
P=0
C
NO
USING SUB. GETNEX()
ACQUIRE 3
NON-LINEAR POINTS
COMPUTE POLYGON PLANE
COEFFICIENTS POLYA(P),
POLYB(P), POLYC(P),
POLYD(P)
ZMIN=0
NETXED=
POLEDG(P)
FIND CLOSEST POINT OF
POLYGON TO VIEWER, IT
WILL BE ONE OF THE
VERTICES
ZMIN=CLOSEST ZS() VALUE
POLZMN(P)=ZMIN
POLZMN(P)
=0
YES
A
NO
OLDP=P
P=POLLNK(P)
A
POLGN=POLGN-1
REMOVE THIS POLYGON
FROM LIST BECAUSE IT
IS EDGE ON TO THE
VIEWPOINT
P=POLLNK(P)
C
B
ORDER POLYGONS IN
POLLNK() WITH CLOSEST
POLYGONS TO VIEWER
LISTED FIRST
RETURN

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     GETNEX: GETS THE INDEX OF THE NEXT EDGE FOR THE CURRENT POLYGON
C        AND THEN FINDS THE TWO END POINTS OF THIS EDGE TO PASS TO THE
C        CALLING SUBROUTINE.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE GETNEX(I)
      COMMON /AAD/ EDGE,EDGEM
      COMMON /AAB/ XS(120),YS(120),ZS(120),POINTM
      COMMON /RA/ EDLINK,POLEDG,NEXTED
      COMMON /RD/ XP,YP,ZP,XN,YN,ZN
      COMMON /DC/ JT1,JT2
      INTEGER EDGE(2,200),EDLINK(200),POLEDG(60),EDGEM,POINTM
      I=0
      IF(NEXTED.EQ.0) RETURN
      JT1=EDGE(1,NEXTED)
      XP=XS(JT1)
      YP=YS(JT1)
      ZP=ZS(JT1)
      JT2=EDGE(2,NEXTED)
      XN=XS(JT2)
      YN=YS(JT2)
      ZN=ZS(JT2)
      NEXTED=EDLINK(NEXTED)
      I=-1
      RETURN
      END
```

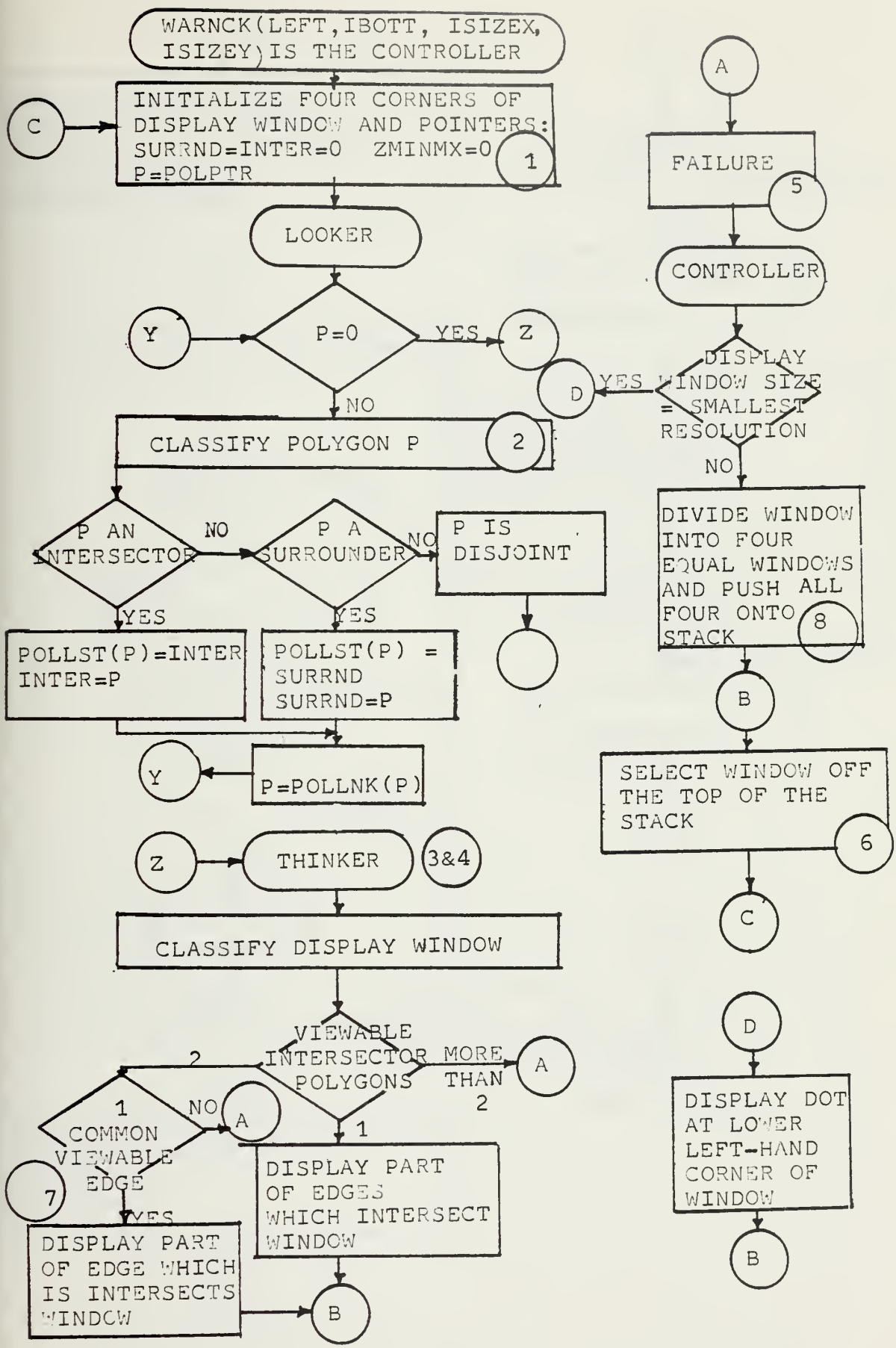

GETNEX(I)
I=0
NEXTED=0
YES
RETURN
NO
JT1=EDGE(1,NEXTED)
JT2=EDGE(2,NEXTED)
XP=XS(JT1)
XN=XS(JT2)
YP=YS(JT1)
YN=YS(JT2)
ZP=ZS(JT1)
ZN=ZS(JT2)
NEXTED=
EDLINK(NEXTED)
I=-1
RETURN

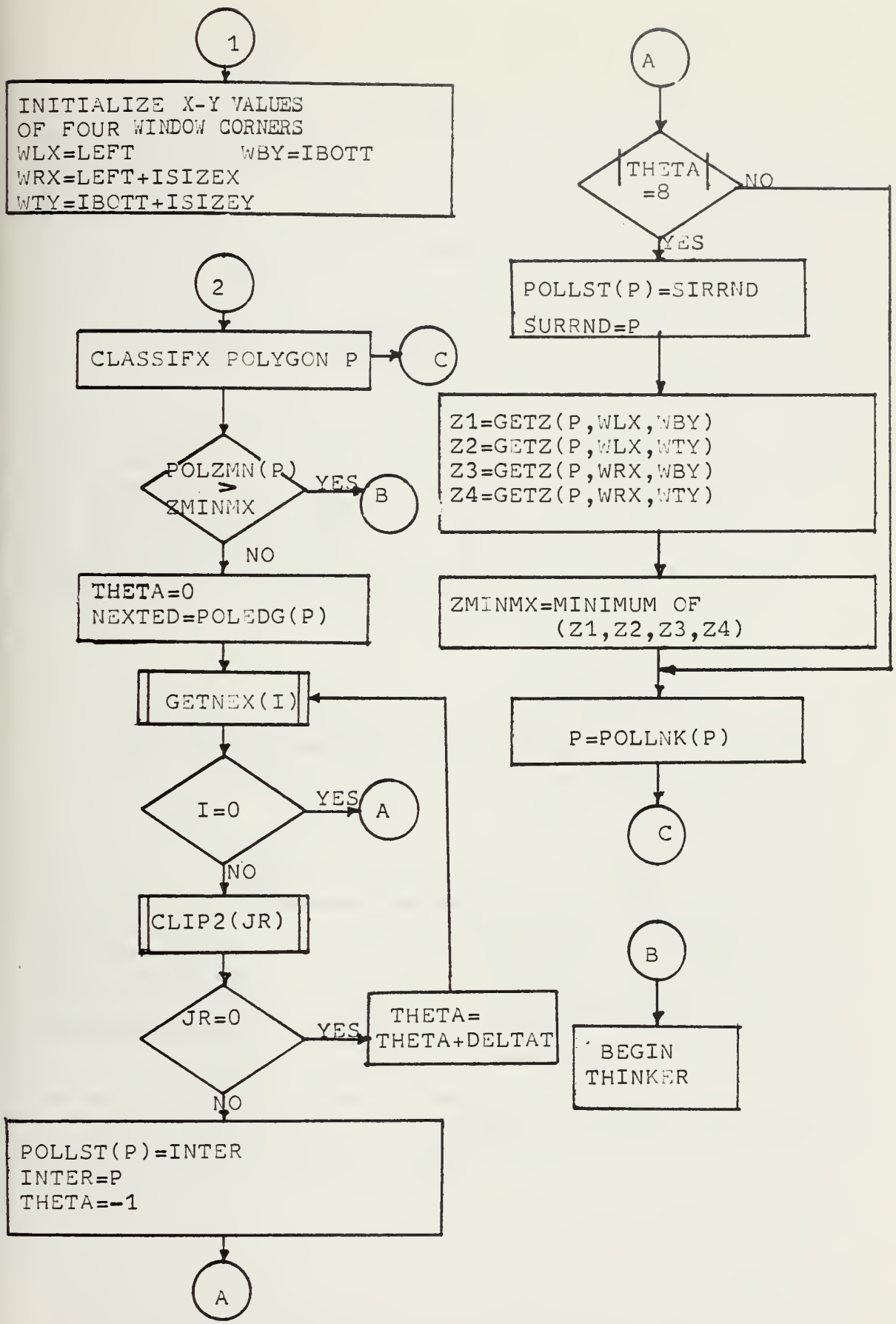
```
CCCC
```

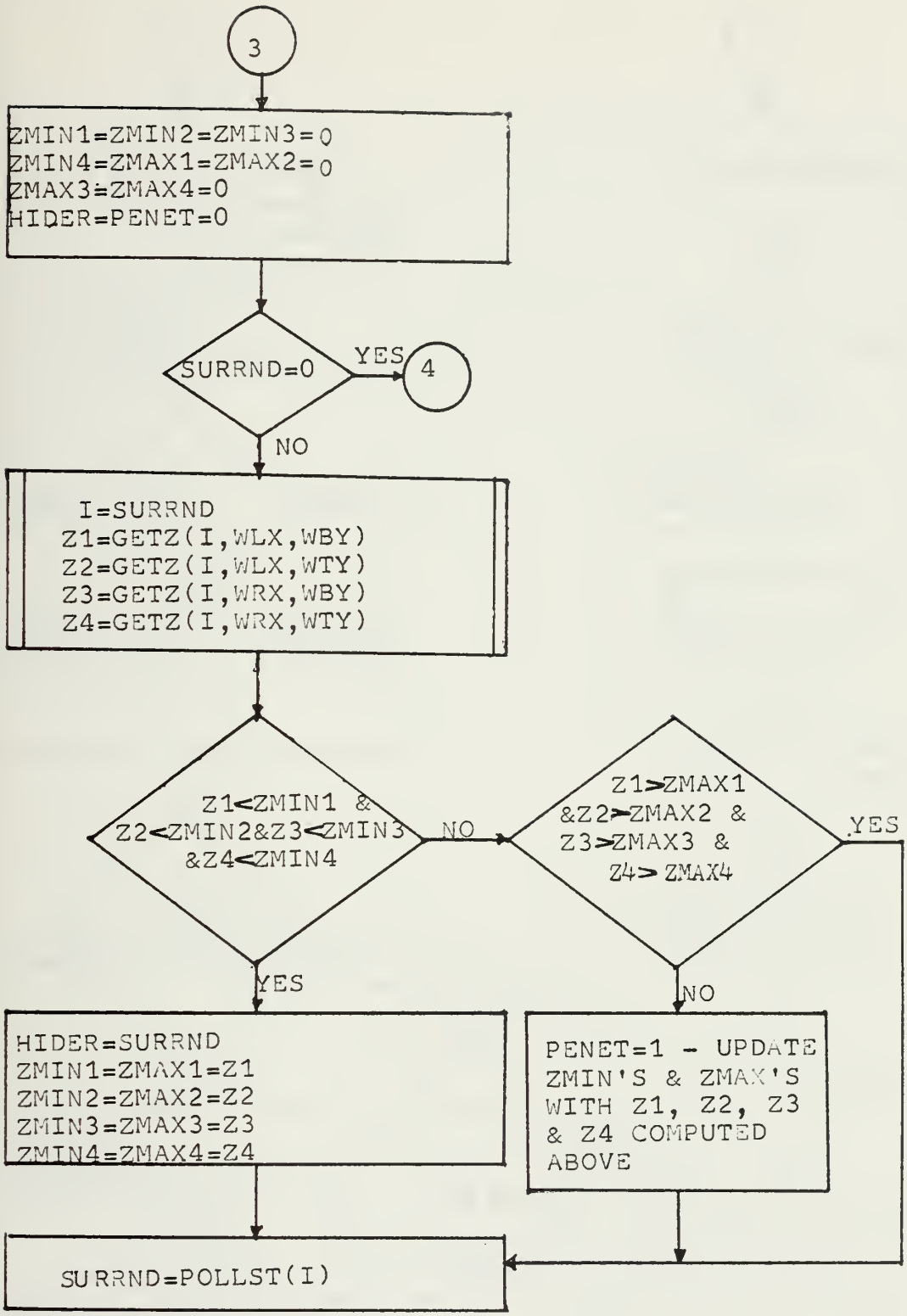
```
      ZMIN4=0.
      ZMAX1=0.
      ZMAX2=0.
      ZMAX3=0.
      ZMAX4=0.
      HIDER=0
      PENET=0
  227 IF(SURRND.EQ.0) GO TO 224
      Z1=GETZ(SURRND,WLX,WBY)
      Z2=GETZ(SURRND,WLX,WTY)
      Z3=GETZ(SURRND,WRX,WBY)
      Z4=GETZ(SURRND,WRX,WTY)
      IF(Z1.GE.ZMIN1) GO TO 225
      IF(Z2.GE.ZMIN2) GO TO 225
      IF(Z3.GE.ZMIN3) GO TO 225
      IF(Z4.GE.ZMIN4) GO TO 225
      HIDER=SURRND
      ZMIN1=Z1
      ZMAX1=Z1
      ZMIN2=Z2
      ZMAX2=Z2
      ZMIN3=Z3
      ZMAX3=Z3
      ZMIN4=Z4
      ZMAX4=Z4
      PENET=0
      GO TO 226
  225 IF(Z1.LE.ZMAX1) GO TO 292
      IF(Z2.LE.ZMAX2) GO TO 292
      IF(Z3.LE.ZMAX3) GO TO 292
      IF(Z4.LE.ZMAX4) GO TO 292
  226 SURRND=POLLST(SURRND)
      GO TO 227
  292 PENET=1
      IF(Z1.LT.ZMIN1)ZMIN1=Z1
      IF(Z1.GT.ZMAX1)ZMAX1=Z1
      IF(Z2.LT.ZMIN2)ZMIN2=Z2
      IF(Z2.GT.ZMAX2)ZMAX2=Z2
      IF(Z3.LT.ZMIN3)ZMIN3=Z3
      IF(Z3.GT.ZMAX3)ZMAX3=Z3
      IF(Z4.LT.ZMIN4)ZMIN4=Z4
      IF(Z4.GT.ZMAX4)ZMAX4=Z4
      GO TO 226
  224 IF(PENET.EQ.1) GO TO 226
      OLDP=0
      P=INTER
 2244 IF(P.EQ.0) GO TO 229
      IF(HIDER.EQ.0) GO TO 229
      Z1=GETZ(P,WLX,WBY)
      Z2=GETZ(P,WLX,WTY)
      Z3=GETZ(P,WRX,WBY)
      Z4=GETZ(P,WRX,WTY)
      IF(Z1.LE.ZMAX1) GO TO 230
      IF(Z2.LE.ZMAX2) GO TO 230
      IF(Z3.LE.ZMAX3) GO TO 230
      IF(Z4.LE.ZMAX4) GO TO 230
      J=POLLST(P)
      IF(OLDP.EQ.0)GO TO 2311
      POLLST(OLDP)=J
      GO TO 231
  230 OLDP=P
      PENET=1
      IF(Z1.GE.ZMIN1) GO TO 228
```

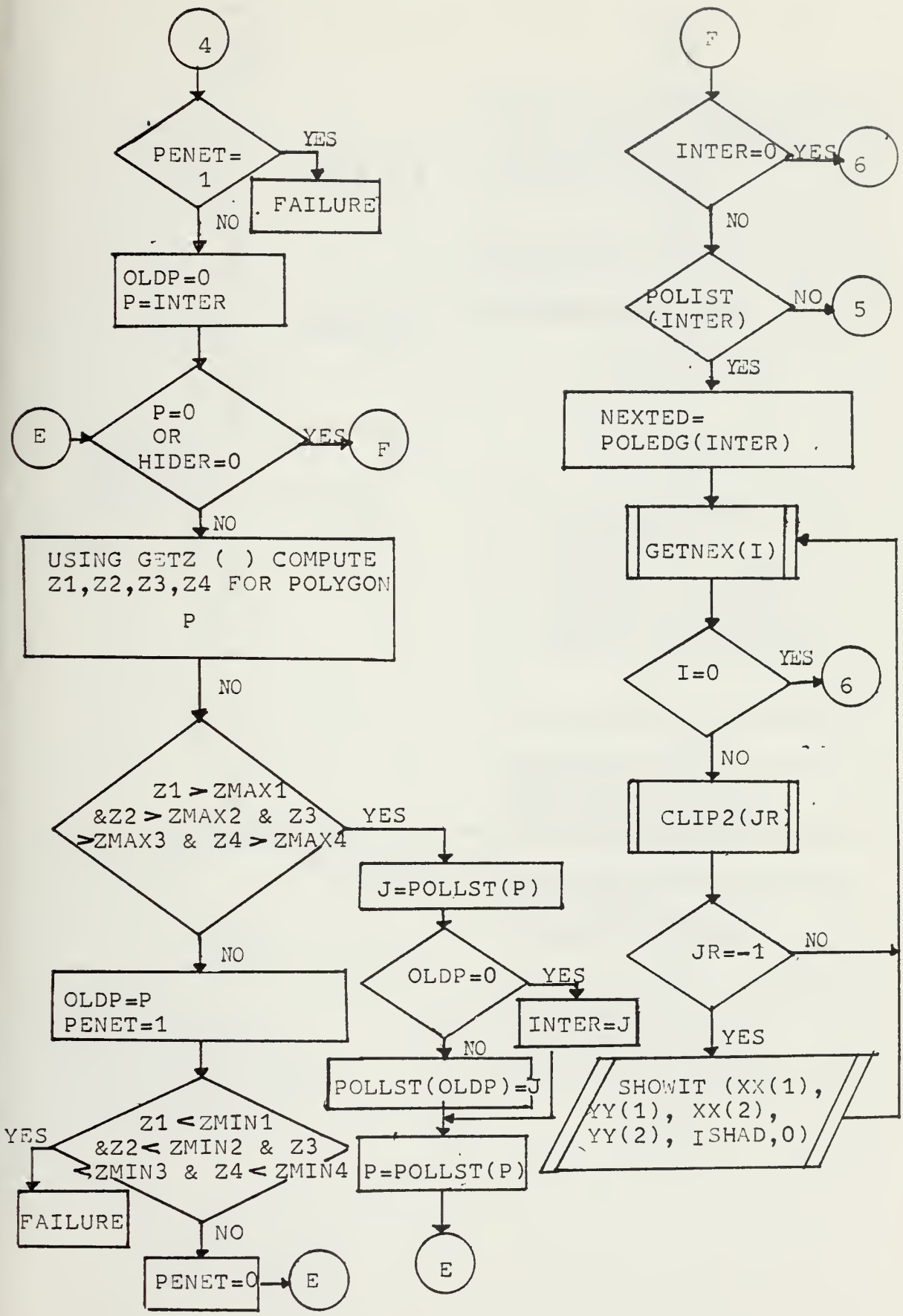
```
      IF(Z2.GE.ZMIN2) GO TO 228
      IF(Z3.GE.ZMIN3) GO TO 228
      IF(Z4.GE.ZMIN4) GO TO 228
      PENET=0
      GO TO 231
 2311 INTER=J
  231 P=POLLST(P)
      GO TO 2244
  229 IF(INTER.EQ.0) GO TO 232
      IF(POLLST(INTER).NE.0) GO TO 228
      NEXTED=POLEDG(INTER)
  233 CALL GETNEX(I)
      IF(I.EQ.0) GO TO 232
      CALL CLIP2(JR)
      IF(JR.EQ.-1) CALL SHOWIT(XX(1),YY(1),XX(2),YY(2),ISHAD,0)
      GO TO 233
 2281 IF(PENET.EQ.1) GO TO 234
      LP=POLLST(INTER)
      IF(POLLST(LP).NE.0) GO TO 234
      NEXTED=POLEDG(INTER)
      IAT=0
 2331 CALL GETNEX(I)
      IF(I.EQ.0) GO TO 2282
      CALL CLIP2(JR)
      IF(JR.EQ.0) GO TO 2331
      IAT=IAT+1
      JJ1=JT1
      JJ2=JT2
      IF(JJ1.GT.JJ2) CALL ISWAP(JJ1,JJ2)
      XT1=XX(1)
      YT1=YY(1)
      XT2=XX(2)
      YT2=YY(2)
      GO TO 2331
 2282 IF(IAT.GT.1) GO TO 234
      IAT=0
      NEXTED=POLEDG(LP)
 2332 CALL GETNEX(I)
      IF(I.EQ.0) GO TO 2283
      CALL CLIP2(JR)
      IF(JR.EQ.0) GO TO 2332
      IAT=IAT+1
      JK1=JT1
      JK2=JT2
      IF(JK1.GT.JK2) CALL ISWAP(JK1,JK2)
      GO TO 2332
 2283 IF(IAT.GT.1) GO TO 234
      IF(JK1.NE.JJ1) GO TO 234
      IF(JK2.NE.JJ2) GO TO 234
      CALL SHOWIT(XT1,YT1,XT2,YT2,ISHAD,0)
      GO TO 232
  228 IF(ISIZEX.GT.1) GO TO 2281
      IF(ISIZEY.GT.1) GO TO 2281
      CALL SHOWIT(WLX,WBY,WLX,WBY,ISHAD,1)
      GO TO 232
  234 ICC=ISIZEX/2
      ICX=ISIZEX-ICC*2
      ICC=ISIZEY/2
      ICY=ISIZEY-ICC*2
      IF(ISIZEX.EQ.1) GO TO 350
      ISIZEX=ISIZEX/2
      IF(ISIZEY.EQ.1) GO TO 360
      ISIZEY=ISIZEY/2
```

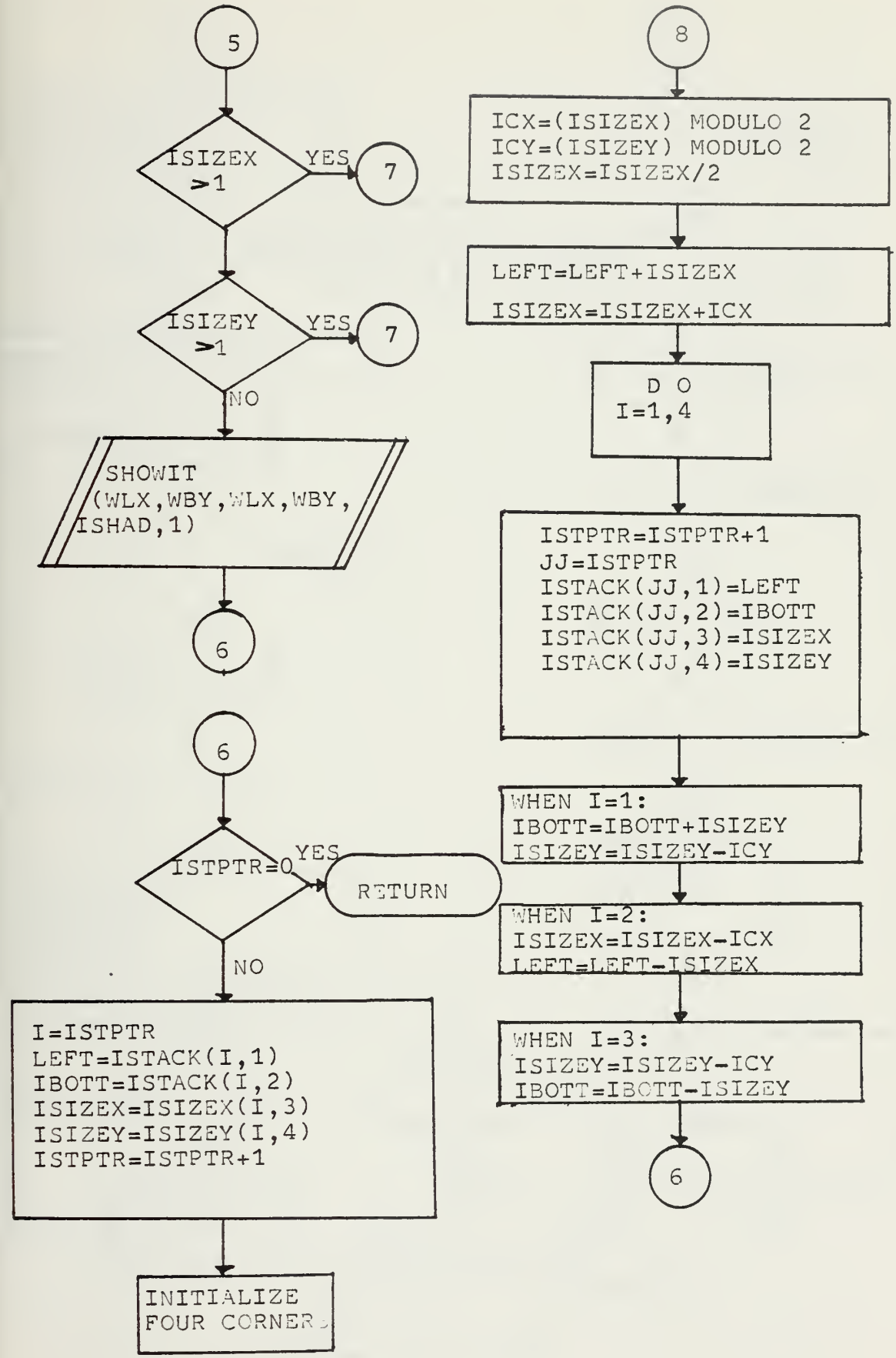
```
      LEFT=LEFT+ISIZEX
      ISIZEX=ISIZEX+ICX
      DO 235 I=1,4
         ISTPTR=ISTPTR+1
         ISTACK(ISTPTR,1)=LEFT
         ISTACK(ISTPTR,2)=IBOTT
         ISTACK(ISTPTR,3)=ISIZEX
         ISTACK(ISTPTR,4)=ISIZEY
         GO TO (236,237,238,235),I
236      IBOTT=IBOTT+ISIZEY
         ISIZEY=ISIZEY+ICY
         GO TO 235
237      ISIZEX=ISIZEX-ICX
         LEFT=LEFT-ISIZEX
         GO TO 235
238      ISIZEY=ISIZEY-ICY
         IBOTT=IBOTT-ISIZEY
235   CONTINUE
232   IF(ISTPTR.LE.0) RETURN
      LEFT=ISTACK(ISTPTR,1)
      IBOTT=ISTACK(ISTPTR,2)
      ISIZEX=ISTACK(ISTPTR,3)
      ISIZEY=ISTACK(ISTPTR,4)
      ISTPTR=ISTPTR-1
      GO TO 239
350   ISIZEY=ISIZEY/2
      IBOTT=IBOTT+ISIZEY
      ISIZEY=ISIZEY+ICY
      DO 351 I=1,2
      ISTPTR=ISTPTR+1
      ISTACK(ISTPTR,1)=LEFT
      ISTACK(ISTPTR,2)=IBOTT
      ISTACK(ISTPTR,3)=ISIZEY
      ISTACK(ISTPTR,4)=ISIZEY
      ISIZEY=ISIZEY-ICY
      IBOTT=IBOTT-ISIZEY
351   CONTINUE
      GO TO 232
360   LEFT=LEFT+ISIZEX
      ISIZEX=ISIZEX+ICX
      DO 361 I=1,2
      ISTPTR=ISTPTR+1
      ISTACK(ISTPTR,1)=LEFT
      ISTACK(ISTPTR,2)=IBOTT
      ISTACK(ISTPTR,3)=ISIZEX
      ISTACK(ISTPTR,4)=ISIZEY
      ISIZEX=ISIZEX-ICX
      LEFT=LEFT-ISIZEX
361   CONTINUE
      GO TO 232
      END
```

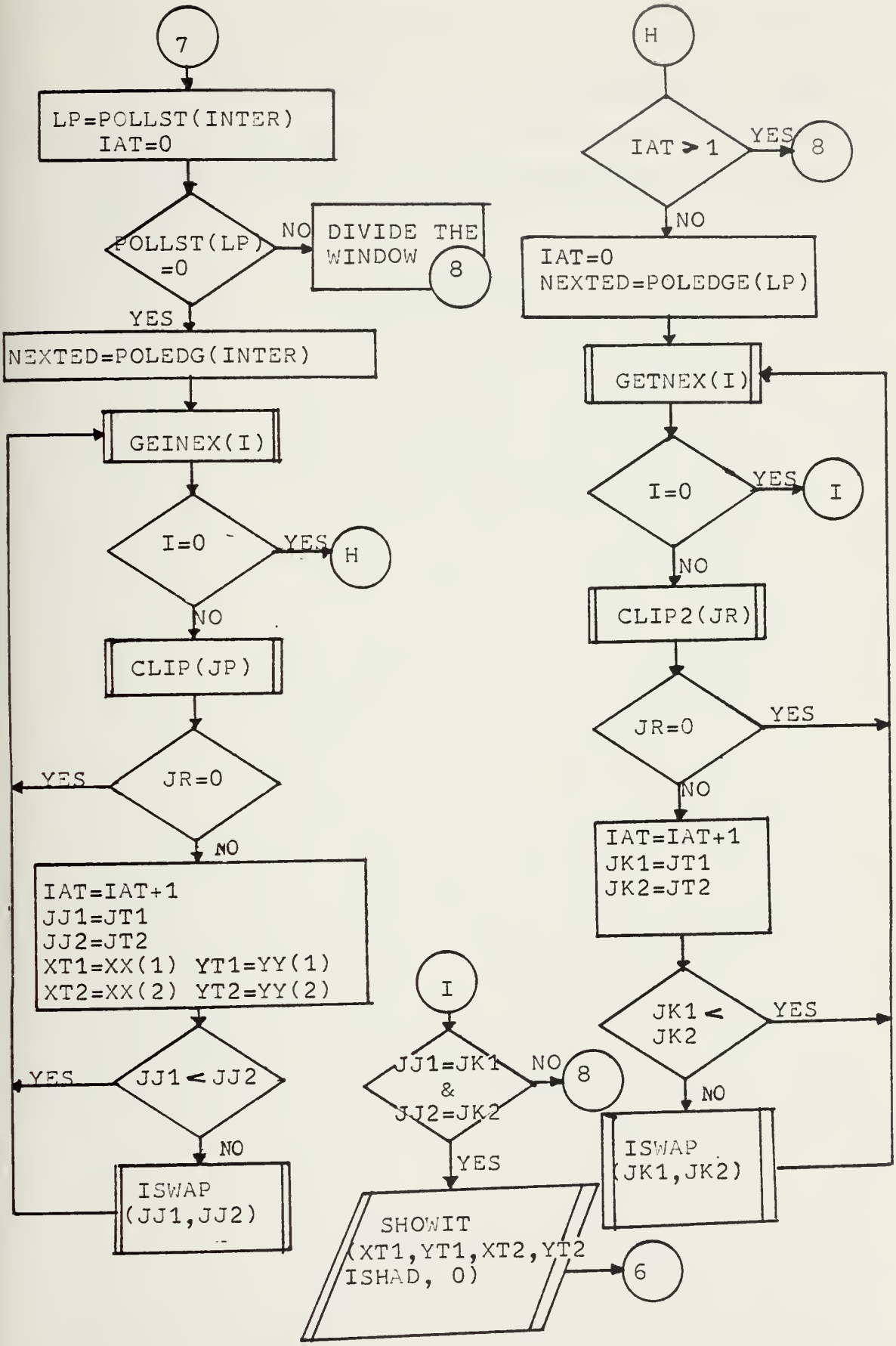

WARNCK(LEFT,IBOTT, ISIZEX, ISIZEY) IS THE CONTROLLER
C
INITIALIZE FOUR CORNERS OF DISPLAY WINDOW AND POINTERS: SURRND=INTER=0 ZMINMX=0 P=POLPTR
1
LOOKER
Y
P=0
YES
Z
NO
CLASSIFY POLYGON P
2
P AN INTERSECTOR
NO
P A SURROUNDER
NO
P IS DISJOINT
YES
YES
POLLST(P)=INTER INTER=P
POLLST(P) = SURRND SURRND=P
Y
P=POLLNK(P)
Z
THINKER
3&4
CLASSIFY DISPLAY WINDOW
VIEWABLE INTERSECTOR POLYGONS
2
MORE THAN 2
A
1
NO
A
1 COMMON VIEWABLE EDGE
7
YES
DISPLAY PART OF EDGE WHICH IS INTERSECTS WINDOW
B
DISPLAY PART OF EDGES WHICH INTERSECT WINDOW
A
FAILURE
5
CONTROLLER
D
YES
DISPLAY WINDOW SIZE = SMALLEST RESOLUTION
NO
DIVIDE WINDOW INTO FOUR EQUAL WINDOWS AND PUSH ALL FOUR ONTO STACK
8
B
SELECT WINDOW OFF THE TOP OF THE STACK
6
C
D
DISPLAY DOT AT LOWER LEFT-HAND CORNER OF WINDOW
B


1
INITIALIZE X-Y VALUES
OF FOUR WINDOW CORNERS
WLX=LEFT WBY=IBOTT
WRX=LEFT+ISIZEX
WTY=IBOTT+ISIZEY
2
CLASSIFX POLYGON P
C
POLZMN(P) > ZMINMX
YES
B
NO
THETA=0
NEXTED=POLEDG(P)
GETNEX(I)
I=0
YES
A
NO
CLIP2(JR)
JR=0
YES
THETA= THETA+DELTAT
NO
POLLST(P)=INTER
INTER=P
THETA=-1
A
A
THETA =8
NO
YES
POLLST(P)=SIRRND
SURRND=P
Z1=GETZ(P,WLX,WBY)
Z2=GETZ(P,WLX,WTY)
Z3=GETZ(P,WRX,WBY)
Z4=GETZ(P,WRX,WTY)
ZMINMX=MINIMUM OF (Z1,Z2,Z3,Z4)
P=POLLNK(P)
C
B
BEGIN THINKER

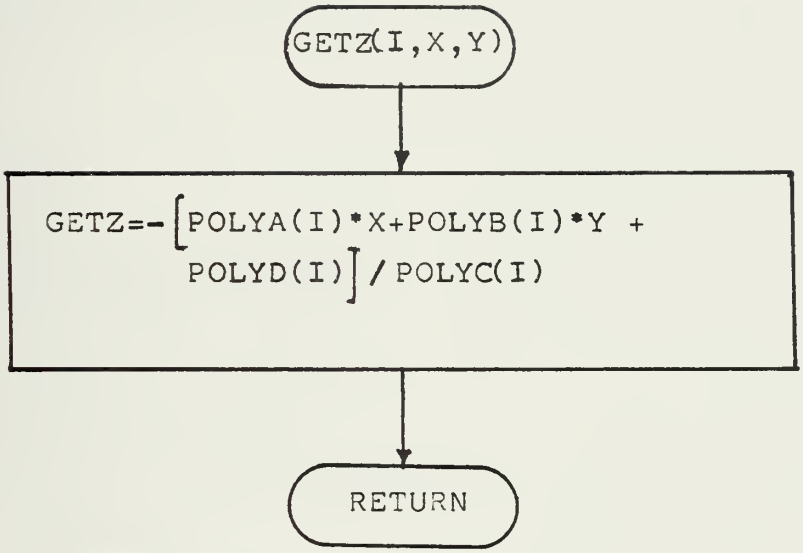

3
ZMIN1=ZMIN2=ZMIN3=0
ZMIN4=ZMAX1=ZMAX2=0
ZMAX3=ZMAX4=0
HIDER=PENET=0
SURRND=0
YES
4
NO
I=SURRND
Z1=GETZ(I,WLX,WBY)
Z2=GETZ(I,WLX,WTY)
Z3=GETZ(I,WRX,WBY)
Z4=GETZ(I,WRX,WTY)
Z1<ZMIN1 &
Z2<ZMIN2&Z3<ZMIN3
&Z4<ZMIN4
NO
Z1>ZMAX1
&Z2>ZMAX2 &
Z3>ZMAX3 &
Z4> ZMAX4
YES
YES
NO
HIDER=SURRND
ZMIN1=ZMAX1=Z1
ZMIN2=ZMAX2=Z2
ZMIN3=ZMAX3=Z3
ZMIN4=ZMAX4=Z4
PENET=1 - UPDATE
ZMIN'S & ZMAX'S
WITH Z1, Z2, Z3
& Z4 COMPUTED
ABOVE
SURRND=POLLST(I)


4
PENET= 1
YES
FAILURE
NO
OLDP=0
P=INTER
E
P=0 OR HIDER=0
YES
F
NO
USING GETZ ( ) COMPUTE Z1,Z2,Z3,Z4 FOR POLYGON P
NO
Z1 > ZMAX1 &Z2 > ZMAX2 & Z3 >ZMAX3 & Z4 > ZMAX4
YES
NO
OLDP=P
PENET=1
YES
Z1 < ZMIN1 &Z2 < ZMIN2 & Z3 <ZMIN3 & Z4 < ZMIN4
FAILURE
NO
PENET=0
E
J=POLLST(P)
OLDP=0
YES
INTER=J
NO
POLLST(OLDP)=J
P=POLLST(P)
E
F
INTER=0
YES
6
NO
POLIST (INTER)
NO
5
YES
NEXTED= POLEDG(INTER)
GETNEX(I)
I=0
YES
6
NO
CLIP2(JR)
JR=-1
NO
YES
SHOWIT (XX(1), YY(1), XX(2), YY(2), ISHAD,0)

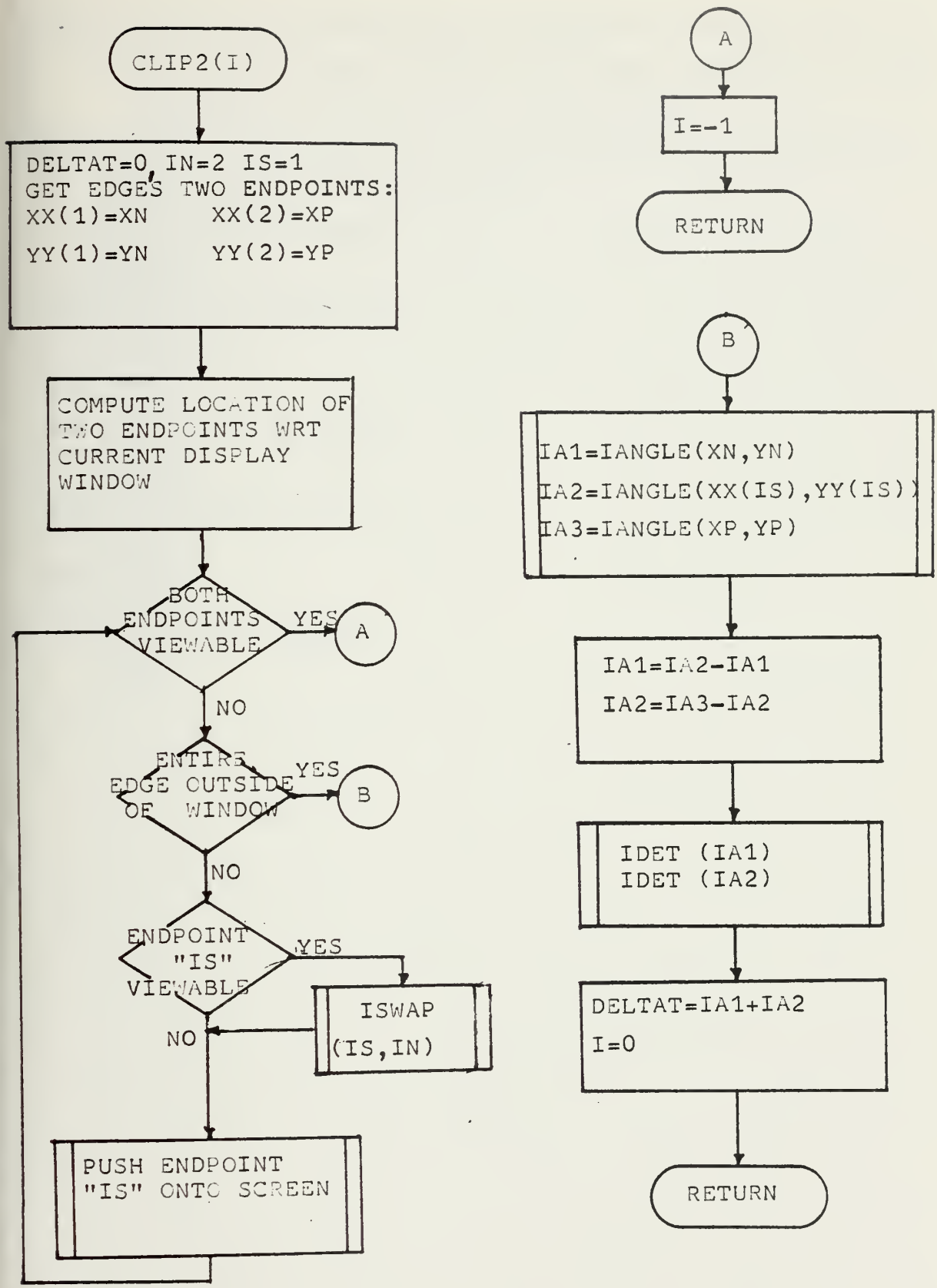

5
ISIZEX >1
YES
7
ISIZEY >1
YES
7
NO
SHOWIT (WLX,WBY,WLX,WBY, ISHAD,1)
6
6
ISTPTR=0
YES
RETURN
NO
I=ISTPTR
LEFT=ISTACK(I,1)
IBOTT=ISTACK(I,2)
ISIZEX=ISIZEX(I,3)
ISIZEY=ISIZEY(I,4)
ISTPTR=ISTPTR+1
INITIALIZE FOUR CORNER
8
ICX=(ISIZEX) MODULO 2
ICY=(ISIZEY) MODULO 2
ISIZEX=ISIZEX/2
LEFT=LEFT+ISIZEX
ISIZEX=ISIZEX+ICX
D O I=1,4
ISTPTR=ISTPTR+1
JJ=ISTPTR
ISTACK(JJ,1)=LEFT
ISTACK(JJ,2)=IBOTT
ISTACK(JJ,3)=ISIZEX
ISTACK(JJ,4)=ISIZEY
WHEN I=1:
IBOTT=IBOTT+ISIZEY
ISIZEY=ISIZEY-ICY
WHEN I=2:
ISIZEX=ISIZEX-ICX
LEFT=LEFT-ISIZEX
WHEN I=3:
ISIZEY=ISIZEY-ICY
IBOTT=IBOTT-ISIZEY
6


7
LP=POLLST(INTER)
IAT=0
POLLST(LP) =0
NO
DIVIDE THE WINDOW
8
YES
NEXTED=POLEDG(INTER)
GEINEX(I)
I=0
YES
H
NO
CLIP(JP)
JR=0
YES
NO
IAT=IAT+1
JJ1=JT1
JJ2=JT2
XT1=XX(1) YT1=YY(1)
XT2=XX(2) YT2=YY(2)
JJ1 < JJ2
YES
NO
ISWAP (JJ1,JJ2)
H
IAT > 1
YES
8
NO
IAT=0
NEXTED=POLEDGE(LP)
GETNEX(I)
I=0
YES
I
NO
CLIP2(JR)
JR=0
YES
NO
IAT=IAT+1
JK1=JT1
JK2=JT2
JK1 < JK2
YES
NO
ISWAP (JK1,JK2)
I
JJ1=JK1 & JJ2=JK2
NO
8
YES
SHOWIT (XT1,YT1,XT2,YT2 ISHAD, 0)
6

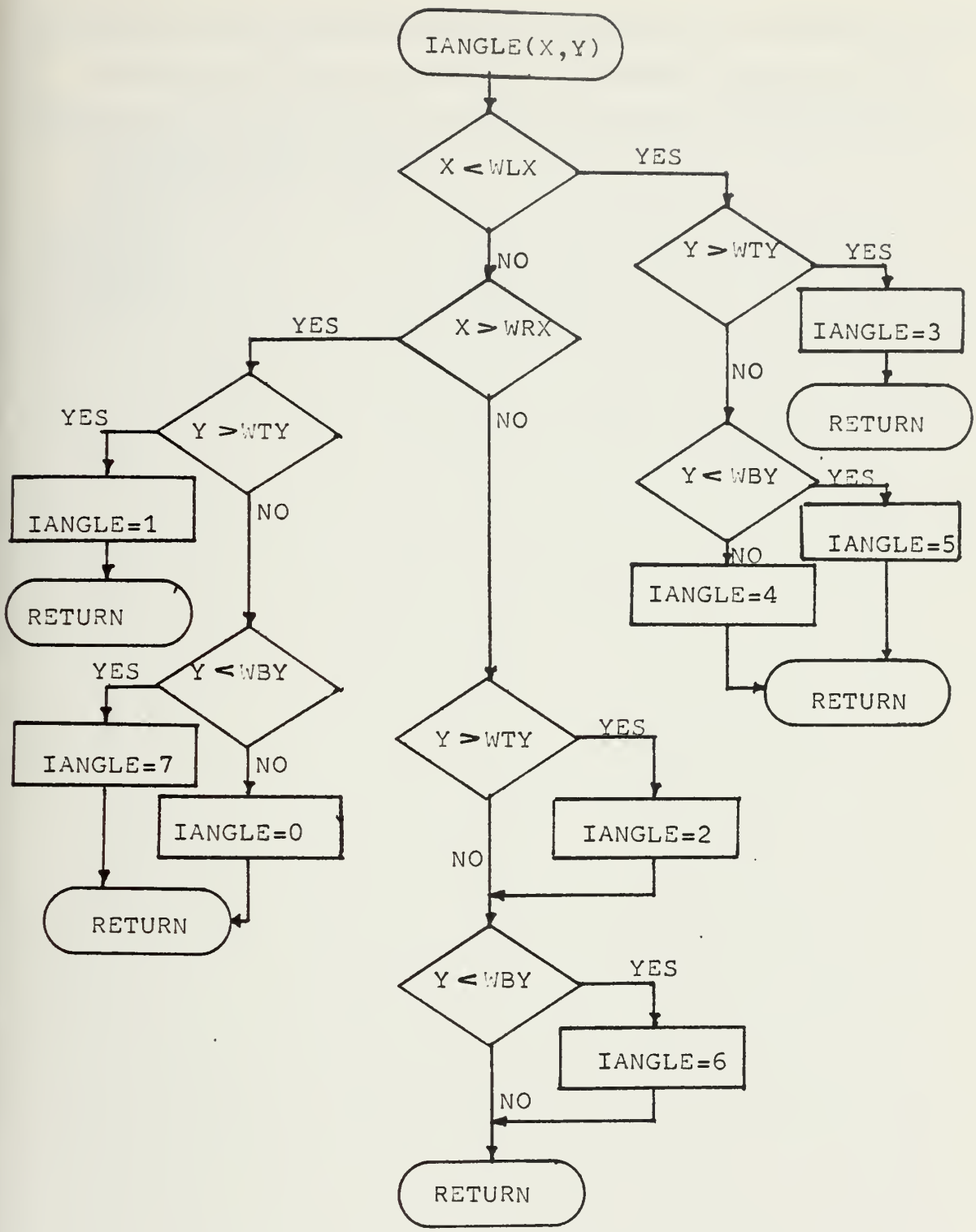
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      GETZ: DETERMINES THE DEPTH OF THE SELECTED POLYGON AT ANY POINT
C         ON THE DISPLAY SCREEN, I.E. THE SREEN COORDINATE Z VALUE.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
       FUNCTION GETZ(I,X,Y)
       COMMON /RC/ POLYA(60),POLYB(60),POLYC(60),POLYD(60),POLZMN(60)
       GETZ=(-POLYA(I)*X-POLYB(I)*Y-POLYD(I))/POLYC(I)
       RETURN
       END
```


GETZ(I,X,Y)
GETZ=-[POLYA(I)*X+POLYB(I)*Y +
POLYD(I)]/POLYC(I)
RETURN

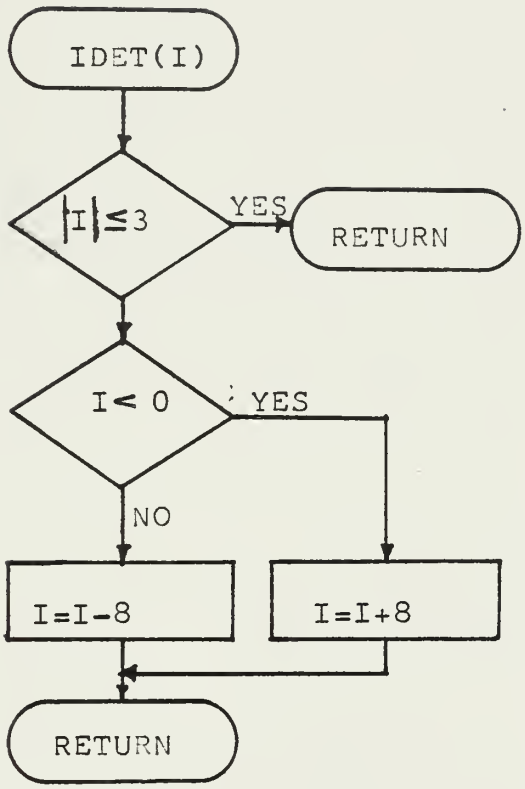
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     CLIP2: CLIPS THE SELECTED EDGE AGAINST THE CURRENT DISPLAY
C        WINDOW IF THE EDGE INTERSECTS THE WINDOW OR COMPUTES THE
C        ANGLE SUBTENDED BY THE EDGE WHICH IS USED TO DETERMINE IF THE
C        CURRENT POLYGON SURROUNDS THIS WINDOW.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE CLIP2(I)
      COMMON /RG/ DELTAT,WLX,WRX,WBY,WTY
      COMMON /RD/XP,YP,ZP,XN,YN,ZN
      COMMON /RH/ XX(2),YY(2)
      COMMON /RI/ ICHK(2,4)
      COMMON /RJ/ IC(2),IS,IN
      INTEGER DELTAT
      DELTAT=0
      IN=2
      IS=1
      XX(1)=XN
      XX(2)=XP
      YY(1)=YN
      YY(2)=YP
      IC(1)=JCODE(XX(1),YY(1),1)
      IC(2)=JCODE(XX(2),YY(2),2)
  257 ICOT=IC(1)+IC(2)
      IF(ICOT.EQ.0) GO TO 250
      DO 55 KK=1,4
      IAAA=ICHK(1,KK)+ICHK(2,KK)
      IF(IAAA.EQ.2) GO TO 251
   55 CONTINUE
      IF(IC(IS).EQ.0) CALL ISWAP(IS,IN)
      IF(ICHK(IS,1).EQ.0) GO TO 253
      CALL PUSH(0,WLX)
      GO TO 257
  253 IF(ICHK(IS,2).EQ.0) GO TO 254
      CALL PUSH(0,WRX)
      GO TO 257
  254 IF(ICHK(IS,3).EQ.0) GO TO 255
      CALL PUSH(1,WBY)
      GO TO 257
  255 CALL PUSH(1,WTY)
      GO TO 257
  250 I=-1
      RETURN
  251 IA1=IANGLE(XN,YN)
      IA2=IANGLE(XX(IS),YY(IS))
      IA3=IANGLE(XP,YP)
      IA1=IA2-IA1
      IA2=IA3-IA2
      CALL IDET(IA1)
      CALL IDET(IA2)
      DELTAT=IA1+IA2
      I=0
      RETURN
      END
```


CLIP2(I)
DELTAT=0, IN=2 IS=1
GET EDGES TWO ENDPOINTS:
XX(1)=XN XX(2)=XP
YY(1)=YN YY(2)=YP
COMPUTE LOCATION OF TWO ENDPOINTS WRT CURRENT DISPLAY WINDOW
BOTH ENDPOINTS VIEWABLE
YES
A
NO
ENTIRE EDGE OUTSIDE OF WINDOW
YES
B
NO
ENDPOINT "IS" VIEWABLE
YES
ISWAP (IS,IN)
NO
PUSH ENDPOINT "IS" ONTO SCREEN
A
I=-1
RETURN
B
IA1=IANGLE(XN,YN)
IA2=IANGLE(XX(IS),YY(IS))
IA3=IANGLE(XP,YP)
IA1=IA2-IA1
IA2=IA3-IA2
IDET (IA1)
IDET (IA2)
DELTAT=IA1+IA2
I=0
RETURN

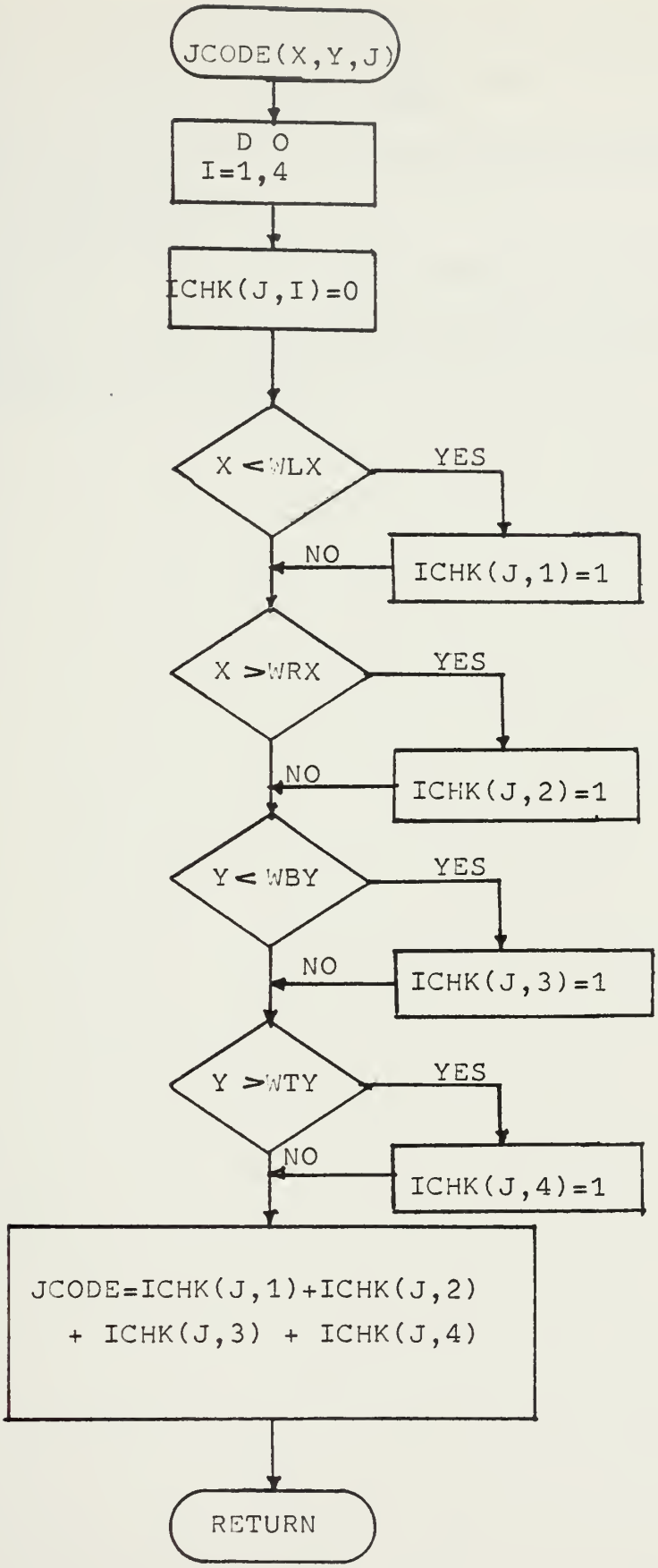
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      IANGLE: USED BY CLIP2 TO DETERMINE THE ANGLE SUBTENDED BY AN EDGE
C         WHICH DOES NOT INTERSECT THE CURRENT WINDOW.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      FUNCTION IANGLE(X,Y)
      COMMON /RG/ DELTAT,WLX,WRX,WBY,WTY
      INTEGER DELTAT
      IF(X.GE.WLX) GO TO 280
      IF(Y.LE.WTY) GO TO 281
      IANGLE=3
      RETURN
  281 IF(Y.GE.WBY) GO TO 282
      IANGLE=5
      RETURN
  282 IANGLE=4
      RETURN
  280 IF(X.LE.WRX) GO TO 283
      IF(Y.LE.WTY) GO TO 284
      IANGLE=1
      RETURN
  284 IF(Y.GE.WBY) GO TO 285
      IANGLE=7
      RETURN
  285 IANGLE=0
      RETURN
  283 IF(Y.GT.WTY) IANGLE=2
      IF(Y.LT.WBY) IANGLE=6
      RETURN
      END
```


IANGLE(X,Y)
X < WLX
YES
NO
Y > WTY
YES
IANGLE=3
RETURN
X > WRX
YES
NO
Y > WTY
YES
IANGLE=1
RETURN
NO
Y < WBY
YES
IANGLE=5
NO
IANGLE=4
RETURN
Y < WBY
YES
IANGLE=7
NO
IANGLE=0
RETURN
Y > WTY
YES
IANGLE=2
NO
Y < WBY
YES
IANGLE=6
NO
RETURN

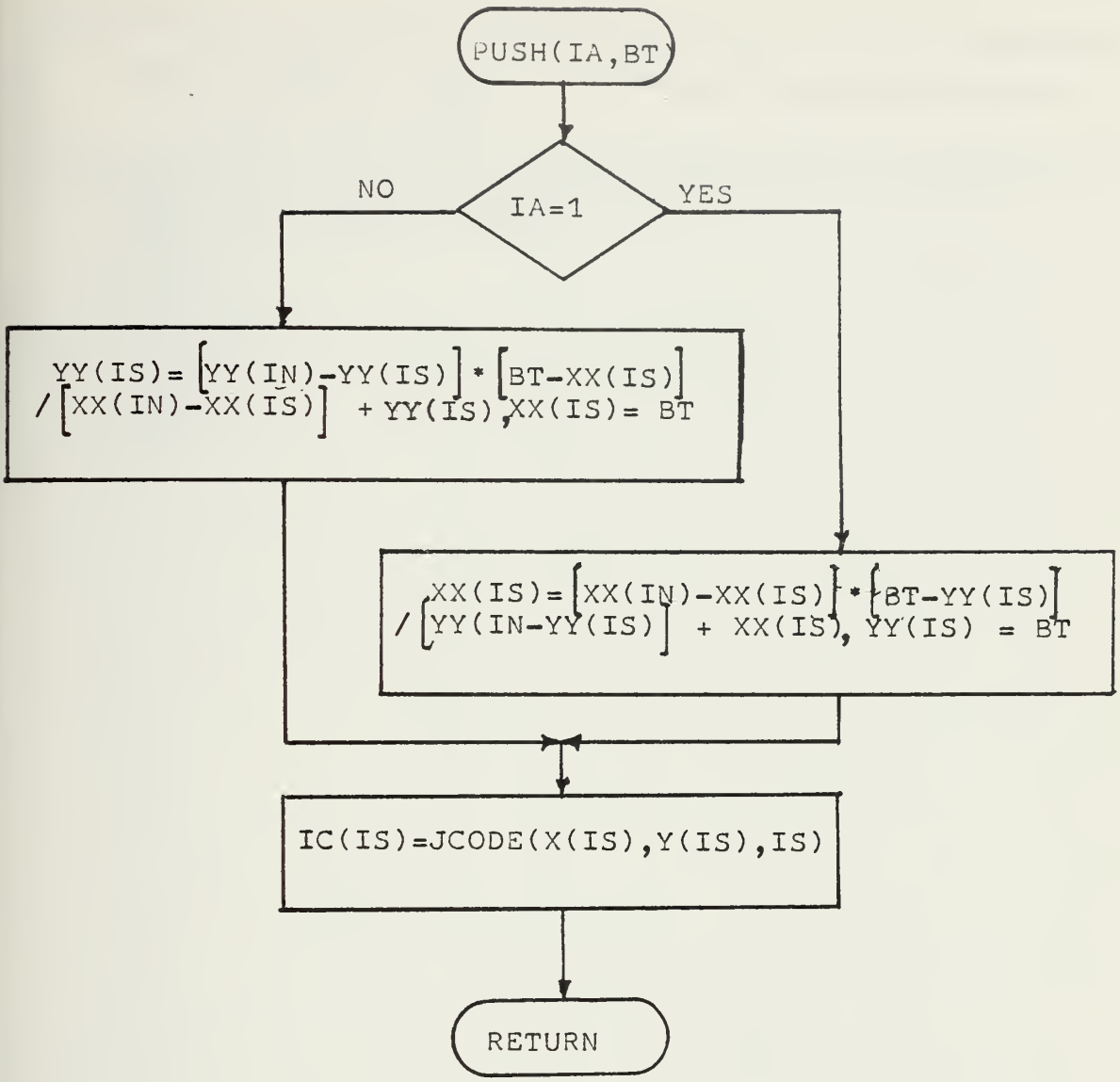
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     IDET: USED BY CLIP2 TO CORRECT THE SUBTENDED ANGLE OF AN EDGE.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE IDET(I)
      IF(IABS(I).LE.3) RETURN
      IF(I.GE.0) GO TO 260
      I=I+8
      RETURN
  260 I=I-8
      RETURN
      END
```


IDET(I)
|I|≤3
YES
RETURN
I< 0
YES
NO
I=I-8
I=I+8
RETURN

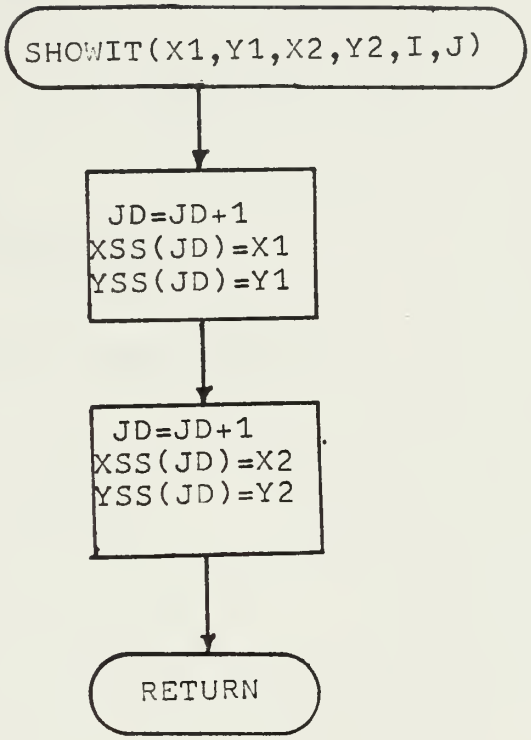
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      JCODE: TO DETERMINE IF AN END POINT OF AN EDGE IS VISIBLE
C         IN THE CURRENT WINDOW AND IF NOT COMPUTES THE POSITION OF THE
C         END POINTS ON THE SCREEN.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      FUNCTION JCODE(X,Y,J)
      COMMON /BG/ DELTAT,WLX,WRX,WBY,WTY
      COMMON /RI/ ICHK(2,4)
      INTEGER DELTAT
      DO 270 I=1,4
         ICHK(J,I)=0
  270 CONTINUE
      IF(X.LT.WLX) ICHK(J,1)=1
      IF(X.GT.WRX) ICHK(J,2)=1
      IF(Y.LT.WBY) ICHK(J,3)=1
      IF(Y.GT.WTY) ICHK(J,4)=1
      JCODE=ICHK(J,1)+ICHK(J,2)+ICHK(J,3)+ICHK(J,4)
      RETURN
      END
```


JCODE(X,Y,J)
D O
I=1,4
ICHK(J,I)=0
X <WLX
YES
NO
ICHK(J,1)=1
X >WRX
YES
NO
ICHK(J,2)=1
Y< WBY
YES
NO
ICHK(J,3)=1
Y >WTY
YES
NO
ICHK(J,4)=1
JCODE=ICHK(J,1)+ICHK(J,2)
+ ICHK(J,3) + ICHK(J,4)
RETURN

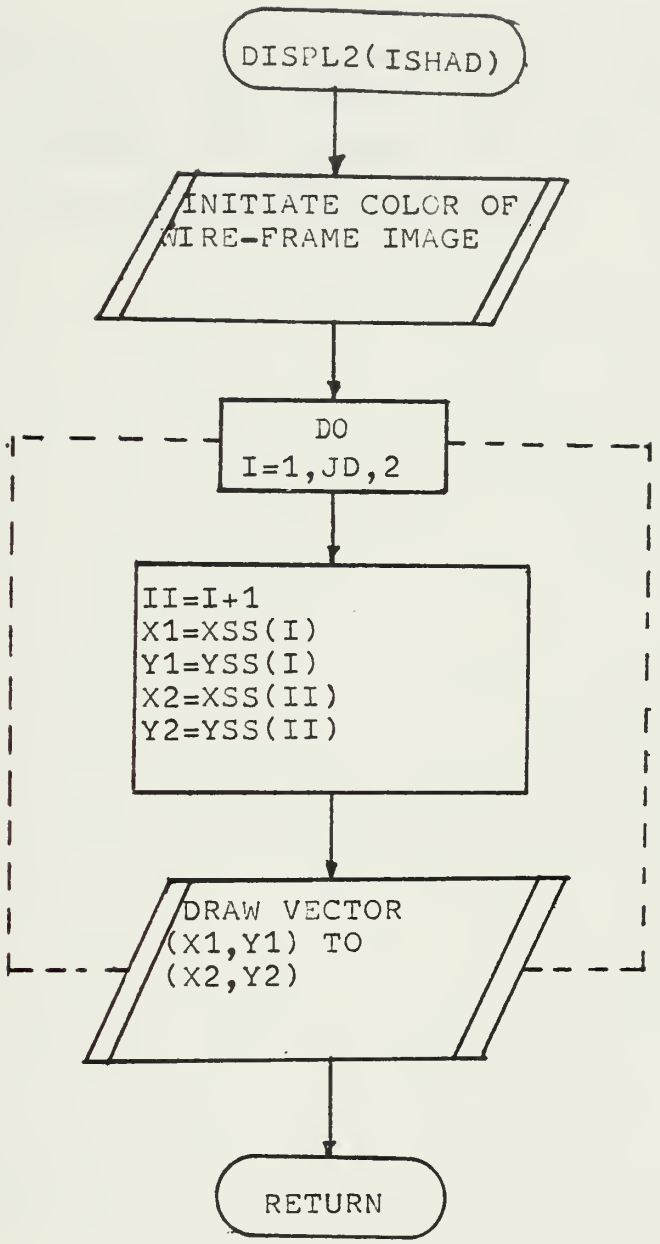
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C     PUSH: USED BY CLIP2 TO DETERMINE THE VISIBLE PORTION OF THIS
C         EDGE IN THE CURRENT DISPLAY WINDOW.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE PUSH(IA,BT)
      COMMON /RH/ XX(2),YY(2)
      COMMON /PJ/ IC(2),IS,IN
      IF(IA.EQ.1) GO TO 275
      YY(IS)=(YY(IN)-YY(IS))*(BT-XX(IS))/(XX(IN)-XX(IS))+YY(IS)
      XX(IS)=BT
      GO TO 276
  275 XX(IS)=(XX(IN)-XX(IS))*(BT-YY(IS))/(YY(IN)-YY(IS))+XX(IS)
      YY(IS)=BT
  276 IC(IS)=JCODE(XX(IS),YY(IS),IS)
      RETURN
      END
```


PUSH(IA,BT)
IA=1
NO
YES
YY(IS)=[YY(IN)-YY(IS)]*[BT-XX(IS)]
/[XX(IN)-XX(IS)] + YY(IS), XX(IS)= BT
XX(IS)=[XX(IN)-XX(IS)]*[BT-YY(IS)]
/[YY(IN-YY(IS)] + XX(IS), YY(IS) = BT
IC(IS)=JCODE(X(IS),Y(IS),IS)
RETURN

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      SHOWIT: STORES THE IMAGE IN THE APPROIATE ARRAYS FOR DISPLAY
C         BY SUBROUTINE DISPLY2.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
       SUBROUTINE SHOWIT(X1,Y1,X2,Y2,I,J)
       COMMON /DA/ XSS(2000),YSS(2000)
       COMMON /DB/ JD
       JD=JD+1
       XSS(JD)=X1
       YSS(JD)=Y1
       JD=JD+1
       XSS(JD)=X2
       YSS(JD)=Y2
       RETURN
       END
```


SHOWIT(X1,Y1,X2,Y2,I,J)
JD=JD+1
XSS(JD)=X1
YSS(JD)=Y1
JD=JD+1
XSS(JD)=X2
YSS(JD)=Y2
RETURN

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      DISPL2 : ACTUALLY DRAWS THE IMAGE WITH THE HIDDEN LINES REMOVED
C         ON THE SELECTED DEVICE.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
       SUBROUTINE DISPL2(ISHAD)
       COMMON /DA/ XSS(2000),YSS(2000)
       COMMON /DB/ JD
       INTEGER VECTOR,COLOR
       DATA EPS,ACC/0.0001,200.0/
         I=COLOR(ISHAD)
       DO 400 I=1,JD,2
          II=I+1
          X1=XSS(I)
          X2=XSS(II)
          Y1=YSS(I)/2.0
          Y2=YSS(II)/2.0
          KI=VECTOR(X1,Y1,X2,Y2)
          IF(KI.LT.0) WRITE(6,3)
  400  CONTINUE
       RETURN
3      FORMAT('THE FUNCTION VECTOR FAILED')
       END
```


DISPL2(ISHAD)
INITIATE COLOR OF WIRE-FRAME IMAGE
DO I=1,JD,2
II=I+1
X1=XSS(I)
Y1=YSS(I)
X2=XSS(II)
Y2=YSS(II)
DRAW VECTOR (X1,Y1) TO (X2,Y2)
RETURN

3. Hidden Surfaces

By calling the subroutine SURFACE, this group was utilized to display the object with its hidden surfaces removed, or to display the hidden or back surfaces, as determined by the calling parameter ILOOK.

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       SURFAC: IS A MASTER SUBROUTINE WHICH CALLS THE
C           SUBROUTINES TO TRANSFORM OBJECT COORDINATES TO SCREEN
C           COORDINATES AND DISPLAYS THE OBJECT WITH SOLID SURFACES.
C           ADDITIONALLY, SURFACE CAN DISPLAY THE SURFACES WHICH
C           ARE CLOSEST TO THE VIEWER OR IT CAN DISPLAY AN OBJECT'S
C            BACK SURFACES.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE SURFAC(ISR,ILOOK)
      COMMON /AAD/EDGE(2,200),EDGEM
      COMMON /AAB/XS(120),YS(120),ZS(120),POINTM
      COMMON /BA/ENILST(200),P(2,200),EDGLST
      COMMON /TA/ IACTVE(60),IFRELS
      COMMON /TB/IYLEFT(60),IYRGHT(60)
      COMMON /TC/IXSLFT(60),IXSRGT(60),SEGFST
      COMMON /TD/XLEFT(60),XRIGHT(60),ZLEFT(60),ZRIGHT(60)
      COMMON /TE/XSPNLF,XSPNRG,XRESL
      COMMON /TF/IBXCNT,IBXTYP
      COMMON /TG/ BXLEFT,BXRGHT,BZLEFT,BZRGHT,BZMIN,BZMAX
      COMMON /TH/SXLEFT,SXRGHT,SZLEFT,SZRGHT
      COMMON /TI/IBSEG1,IBSEG2,DIV,SDIV,IBFULL,ISFULL
      COMMON /TJ/SEGSAM,IMPLS1,IMPLS2,PREVIS
      COMMON /TK/POLSEG(60),POLGON(60)
      COMMON /TL/DXLEFT(60),DXRGHT(60)
      COMMON /TM/SAMFRE,SAMLST,SAMLNK(120),SAMX(120)
      COMMON /TN/IMPLFT,XLSTSM
      COMMON /TO/SEGCNT,LSTSEG,IY
      COMMON /AB/POLYGN(60,11),POLGN,SHAD(60)
      COMMON /JJ/VSX,VSY,VCX,VCY
      DIMENSION IYENTR(240),DZLEFT(60),DZRGHT(60),CHANGE(60)
      DIMENSION SEGLST(60),ISR(2)
      INTEGER POLYGN,POLGN,SHAD,SEGLST,POLSEG,CHANGE,CHG,SEG,SEG1
      INTEGER EDGE,EDGEM,ENILST,SEGFST,SAMFRE,PTR,P,PI,POLCNG,POLGON
      INTEGER POINTM,EDGLST,SEGSAM,SEGCNT,SEGACT,SEGLO,SEGOUT,SAMLNK
      INTEGER SAMFST,SAMLST,SAMPLE,CURSEG,PREVIS,SAMX
      DATA MAXSEG,APS/60,0.0001/,SAMFST/0/
      CALL RDYCLP
      CALL CLIP
      CALL SCRN
      CALL INIT2(ISR)
      IYRES=2.0*VSY+APS
      CALL SHOWIN(IYRES)
      DO 101 I=1,IYRES
101   IYENTR(I)=0
      XRESL=2.0*VSX
      IXRES=XRESL+APS
      MM=MAXSEG-1
      IACTVE(MAXSEG)=0
      DO 53 I=1,MM
53    IACTVE(I)=I+1
      MM=MAXSEG*2
      SAMLNK(MM)=0
      MM=MM-1
      DO 54 I=1,MM
54    SAMLNK(I)=I+1
      IFRELS=1
      SAMFRE=1
      IMPLS1=0
      IMPLS2=0
      SEGFST=0
      PTR=EDGLST
```

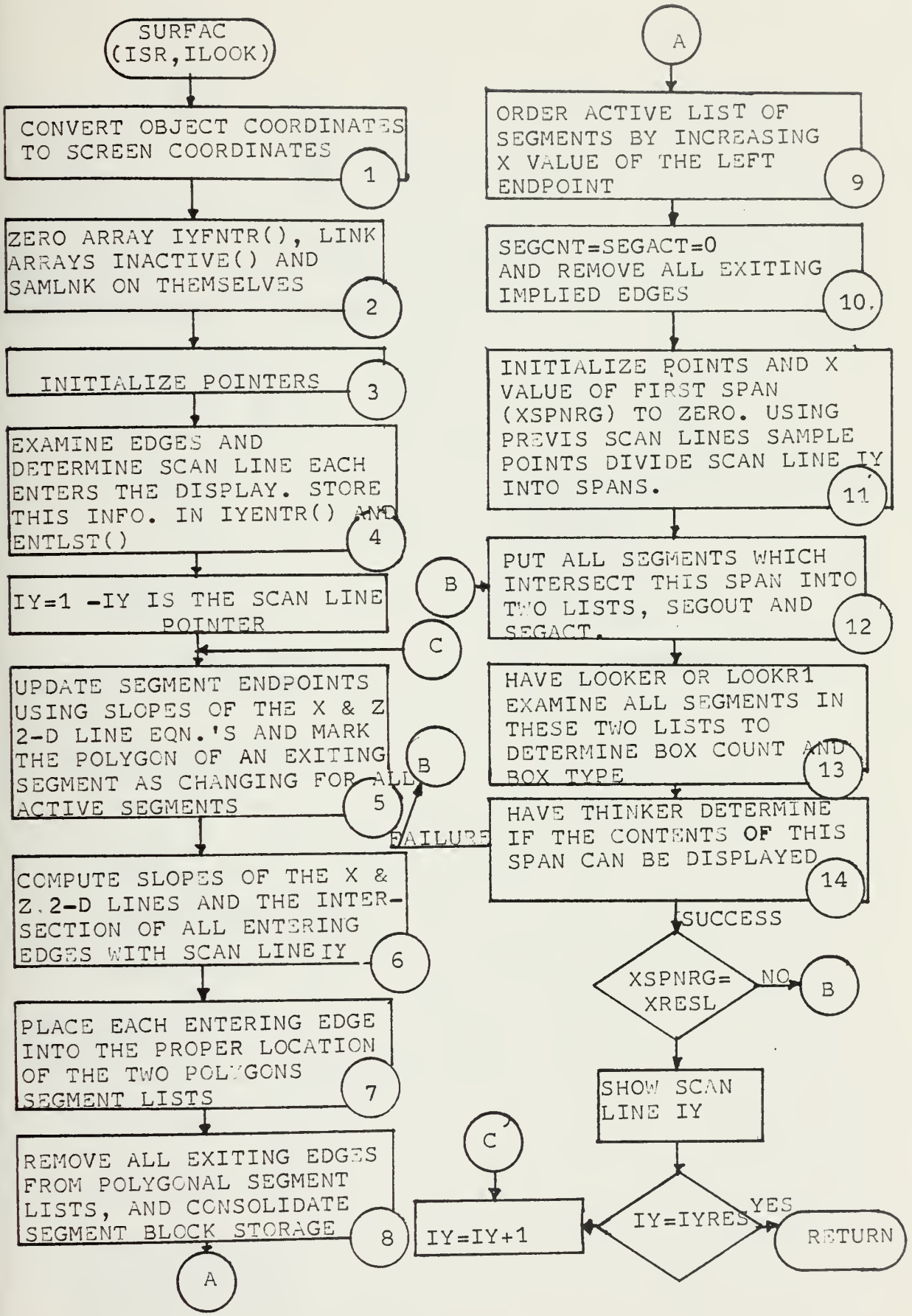
```
56    IF(PTR.EQ.0) GO TO 55
      NEXT=ENTLST(PTR)
      I=P(1,PTR)
      IF(I.NE.0.AND.SHAD(I).NE.0) GO TO 57
      I=P(2,PTR)
      IF(I.NE.0.AND.SHAD(I).NE.0) GO TO 57
      PTR=NEXT
      GO TO 56
57    J=EDGE(1,PTR)
      K=EDGE(2,PTR)
      IF(YS(J).LE.YS(K)) GO TO 58
      CALL ISWAP(EDGE(1,PTR),EDGE(2,PTR))
      J=K
58    I=YS(J)+0.99999
      IF(I.LT.1.OR.I.GT.IYRES) GO TO 100
      ENTLST(PTR)=IYENTR(I)
      IYENTR(I)=PTR
      PTR=NEXT
      GO TO 56
55      DO 59 IY=1,IYRES
        Y=IY
        POLCHG=-1
        SEG=SEGFST
64      IF(SEG.EQ.0) GO TO 60
        XLEFT(SEG)=XLEFT(SEG)+DXLEFT(SEG)
        XRIGHT(SEG)=XRIGHT(SEG)+DXRGHT(SEG)
        ZLEFT(SEG)=ZLEFT(SEG)+DZLEFT(SEG)
        ZRIGHT(SEG)=ZRIGHT(SEG)+DZRGHT(SEG)
        IY1=IYLEFT(SEG)+1
        IY2=IYRGHT(SEG)+1
        IYLEFT(SEG)=IY1
        IYRGHT(SEG)=IY2
        IF(IY1.NE.0.AND.IY2.NE.0) GO TO 61
        PTR=POLGON(SEG)
        IF(PTR.EQ.0) GO TO 62
        IF(CHANGE(PTR).NE.0) GO TO 61
        CHANGE(PTR)=POLCHG
        POLCHG=PTR
        GO TO 61
62      CALL RMXSRT(SEG)
        CALL RETBLK(SEG)
61      SEG=IXSRGT(SEG)
        GO TO 64
60      PTR=IYENTR(IY)
65      IF(PTR.EQ.0) GO TO 66
        IVV1=EDGE(1,PTR)
        IVV2=EDGE(2,PTR)
        IYFRST=YS(IVV1)
        IYLAST=YS(IVV2)
        IDELY=IYFRST-IYLAST
        REALDY=YS(IVV2)-YS(IVV1)
        IF(IDELY.GE.0) GO TO 67
        XSLOPE=(XS(IVV2)-XS(IVV1))/REALDY
        XFIRST=XS(IVV1)+XSLOPE*(Y-YS(IVV1))
        ZSLOPE=(ZS(IVV2)-ZS(IVV1))/REALDY
        ZFIRST=ZS(IVV1)+ZSLOPE*(Y-YS(IVV1))
        DO 68 I=1,2
            PI=P(I,PTR)
            IF(PI.EQ.0) GO TO 68
            IF(CHANGE(PI).NE.0) GO TO 69
            CHANGE(PI)=POLCHG
            POLCHG=PI
69          SEG=SEGLST(PI)
```

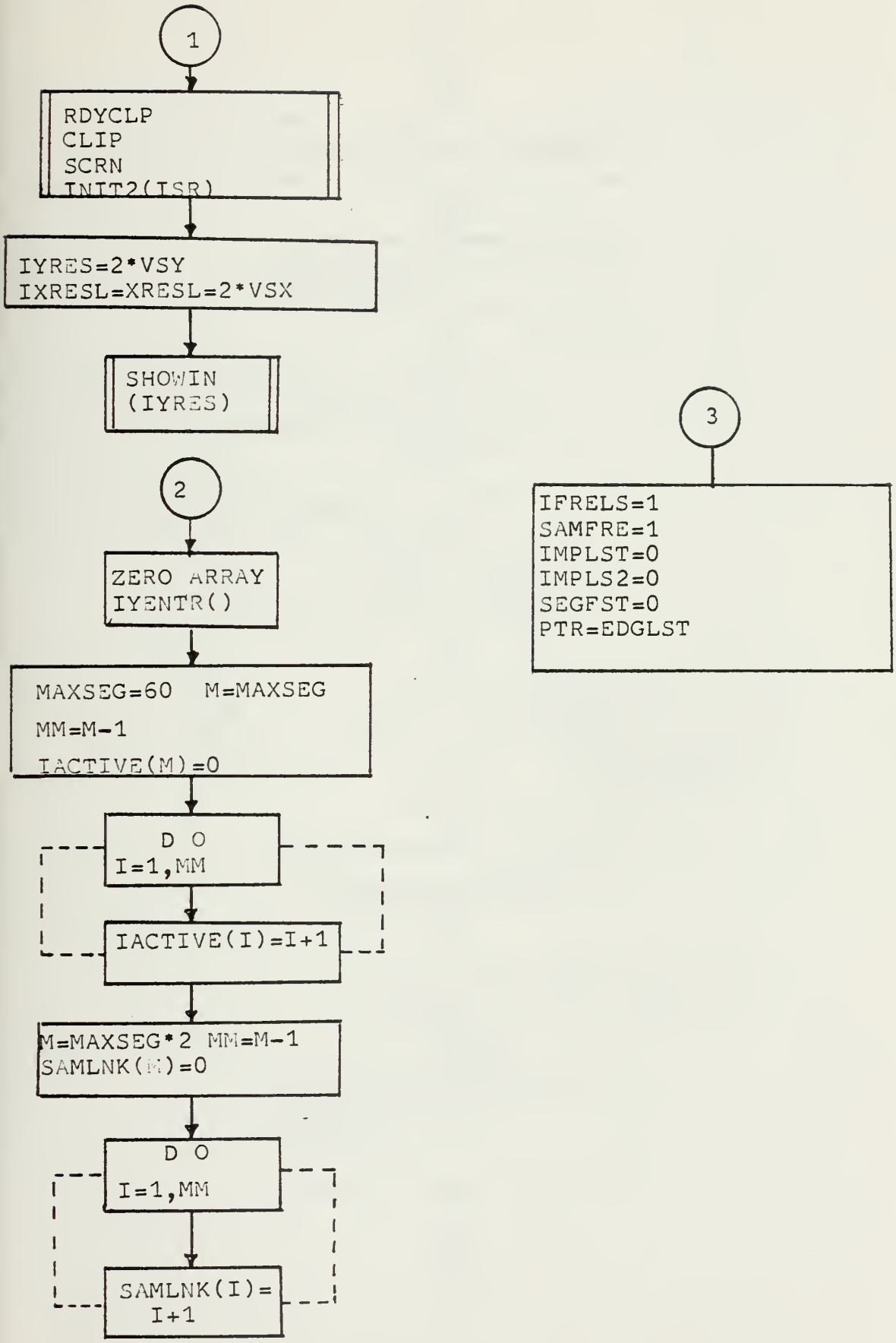
```
            PREVIS=0
71          IF(SEG.EQ.0) GO TO 70
            ITE1=0
            IF((XFIRST.LT.XLEFT(SEG)).OR.((XFIRST.EQ.XLEFT(SEG)).AND.
     &          (XSLOPE.LT.DXLEFT(SEG)))) ITE1=8
            ITE2=0
            IF((XFIRST.LT.XRIGHT(SEG)).OR.((XFIRST.EQ.XRIGHT(SEG)).AND.
     &          (XSLOPE.LT.DXRGHT(SEG)))) ITE2=4
            IY1=0
            IF(IYLEFT(SEG).LT.0)IY1=2
            IY2=0
            IF(IYRGHT(SEG).LT.0)IY2=1
            II=ITE1+ITE2+IY1+IY2+1
            GO TO (3,3,3,3,3,70,3,72,3,3,70,72,3,70,70,70),II
3           PREVIS=SEG
            SEG=POLSEG(SEG)
            GO TO 71
70          SEG1=IGTBLK(IR)
            IF(SEG1.EQ.0) GO TO 160
            POLGON(SEG1)=PI
            XLEFT(SEG1)=XFIRST
            DXLEFT(SEG1)=XSLOPE
            ZLEFT(SEG1)=ZFIRST
            DZLEFT(SEG1)=ZSLOPE
            IYLEFT(SEG1)=IDELY
            CALL PUTXST(SEG1)
            IF(PREVIS.EQ.0) GO TO 73
            POLSEG(PREVIS)=SEG1
            GO TO 74
73          SEGLST(PI)=SEG1
74          POLSEG(SEG1)=SEG
            GO TO 68
72          SEG1=IGTBLK(IR)
            IF(SEG1.EQ.0) GO TO 160
            POLGON(SEG1)=PI
            XLEFT(SEG1)=XLEFT(SEG)
            DXLEFT(SEG1)=DXLEFT(SEG)
            ZLEFT(SEG1)=ZLEFT(SEG)
            DZLEFT(SEG1)=DZLEFT(SEG)
            IYLEFT(SEG1)=IYLEFT(SEG)
            IYLEFT(SEG)=0
            XRIGHT(SEG1)=XFIRST
            DXRGHT(SEG1)=XSLOPE
            ZRIGHT(SEG1)=ZFIRST
            DZRGHT(SEG1)=ZSLOPE
            IYRGHT(SEG1)=IDELY
            CALL PUTXST(SEG1)
            IF(PREVIS.EQ.0) GO TO 608
            POLSEG(PREVIS)=SEG1
            GO TO 609
608         SEGLST(PI)=SEG1
609         POLSEG(SEG1)=SEG
            PREVIS=SEG1
68        CONTINUE
67      PTR=ENTLST(PTR)
          GO TO 65
93       PREVIS=SEG
         SEG=POLSEG(SEG)
          GO TO 75
66      IF(POLCHG.EQ.-1) GO TO 75
        PI=POLCHG
        POLCHG=CHANGE(PI)
        CHANGE(PI)=0
```

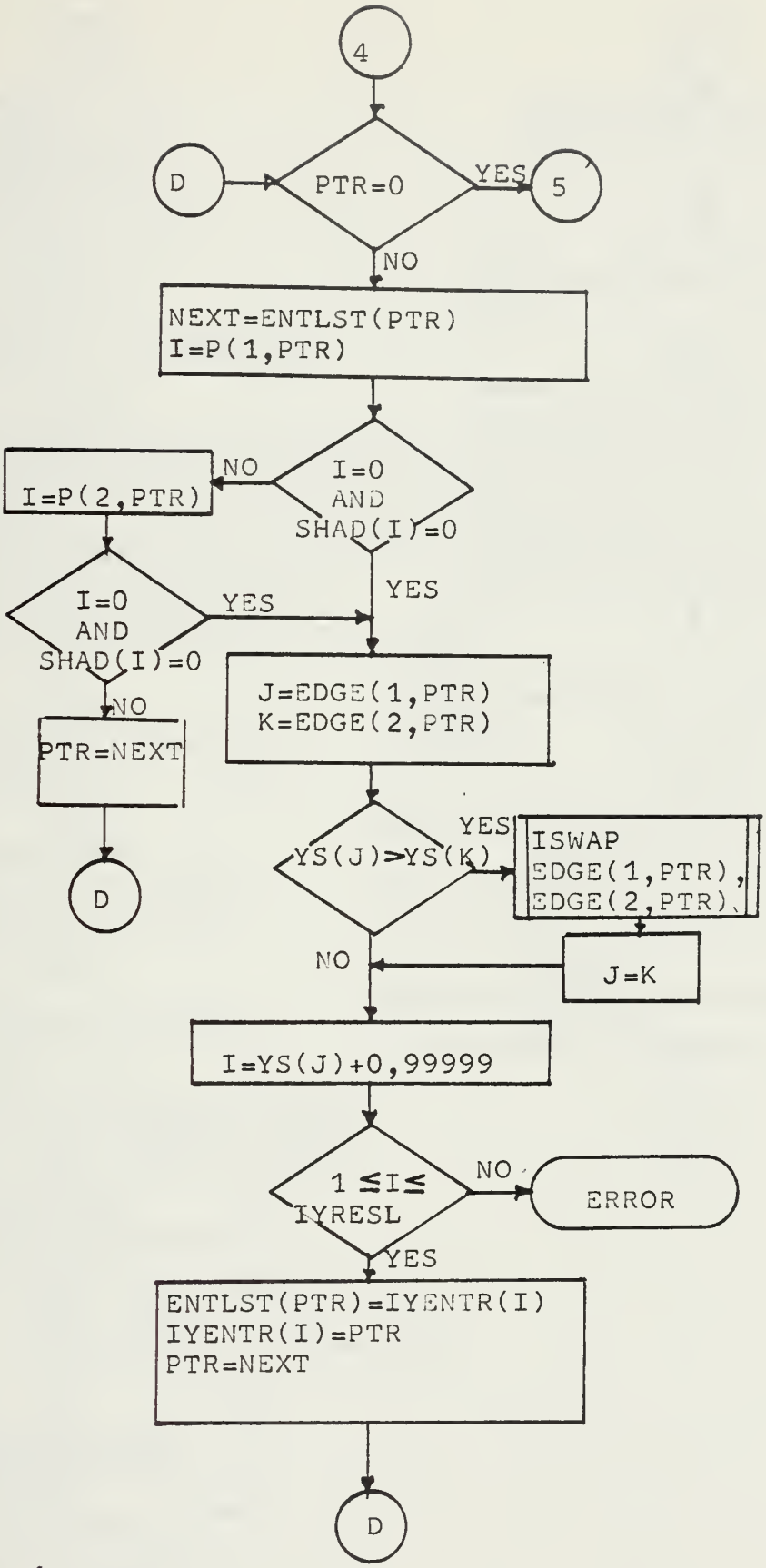
```
      PREVIS=0
      SEG=SEGLST(PI)
76    IF(SEG.EQ.0) GO TO 66
      IY1=IYLEFT(SEG)
      IY2=IYRGHT(SEG)
      IF(IY1.LT.0.AND.IY2.LT.0) GO TO 93
      IF(IY1.EQ.0.AND.IY2.EQ.0) GO TO 77
      IF(IY1.NE.0.OR.IY2.GE.0) GO TO 78
      IYLEFT(SEG)=IYRGHT(SEG)
      IYRGHT(SEG)=0
      XLEFT(SEG)=XRIGHT(SEG)
      DXLEFT(SEG)=DXRGHT(SEG)
      ZLEFT(SEG)=ZRIGHT(SEG)
      DZLEFT(SEG)=DZRGHT(SEG)
      GO TO 76
77    I=POLSEG(SEG)
      IF(PREVIS.EQ.0) GO TO 79
      POLSEG(PREVIS)=I
      GO TO 80
79    SEGLST(PI)=I
80    CALL RMXSRT(SEG)
      CALL RETBLK(SEG)
      SEG=I
      GO TO 76
78    NEXT=POLSEG(SEG)
      IF(NEXT.EQ.0) GO TO 110
     IF(IYLEFT(NEXT).GE.0) GO TO 81
      IYRGHT(SEG)=IYLEFT(NEXT)
      IYLEFT(NEXT)=0
      XRIGHT(SEG)=XLEFT(NEXT)
      DXRGHT(SEG)=DXLEFT(NEXT)
      ZRIGHT(SEG)=ZLEFT(NEXT)
      DZRGHT(SEG)=DZLEFT(NEXT)
      GO TO 76
81    IF(IYRGHT(NEXT).GE.0) GO TO 82
      IYRGHT(SEG)=IYRGHT(NEXT)
      IYRGHT(NEXT)=0
      XRIGHT(SEG)=XRIGHT(NEXT)
      DXRGHT(SEG)=DXRGHT(NEXT)
      ZRIGHT(SEG)=ZRIGHT(NEXT)
      DZRGHT(SEG)=DZRGHT(NEXT)
      GO TO 76
82    POLSEG(SEG)=POLSEG(NEXT)
      CALL RMXSRT(NEXT)
      CALL RETBLK(NEXT)
      GO TO 76
75    CHG=0
      SEG=SEGFST
84    IF(SEG.EQ.0) GO TO 83
      I=IXSRGT(SEG)
      IF(I.EQ.0) GO TO 83
      IF(XLEFT(SEG).LE.XLEFT(I)) GO TO 85
      CHG=1
      K=IXSLFT(SEG)
      IF(K.NE.0) IXSRGT(K)=I
      IXSLFT(I)=K
      IXSLFT(SEG)=I
      K=IXSRGT(I)
      IF(K.NE.0)IXSLFT(K)=SEG
      IXSRGT(SEG)=K
      IXSRGT(I)=SEG
      IF(SEGFST.EQ.SEG) SEGFST=I
      GO TO 84
```

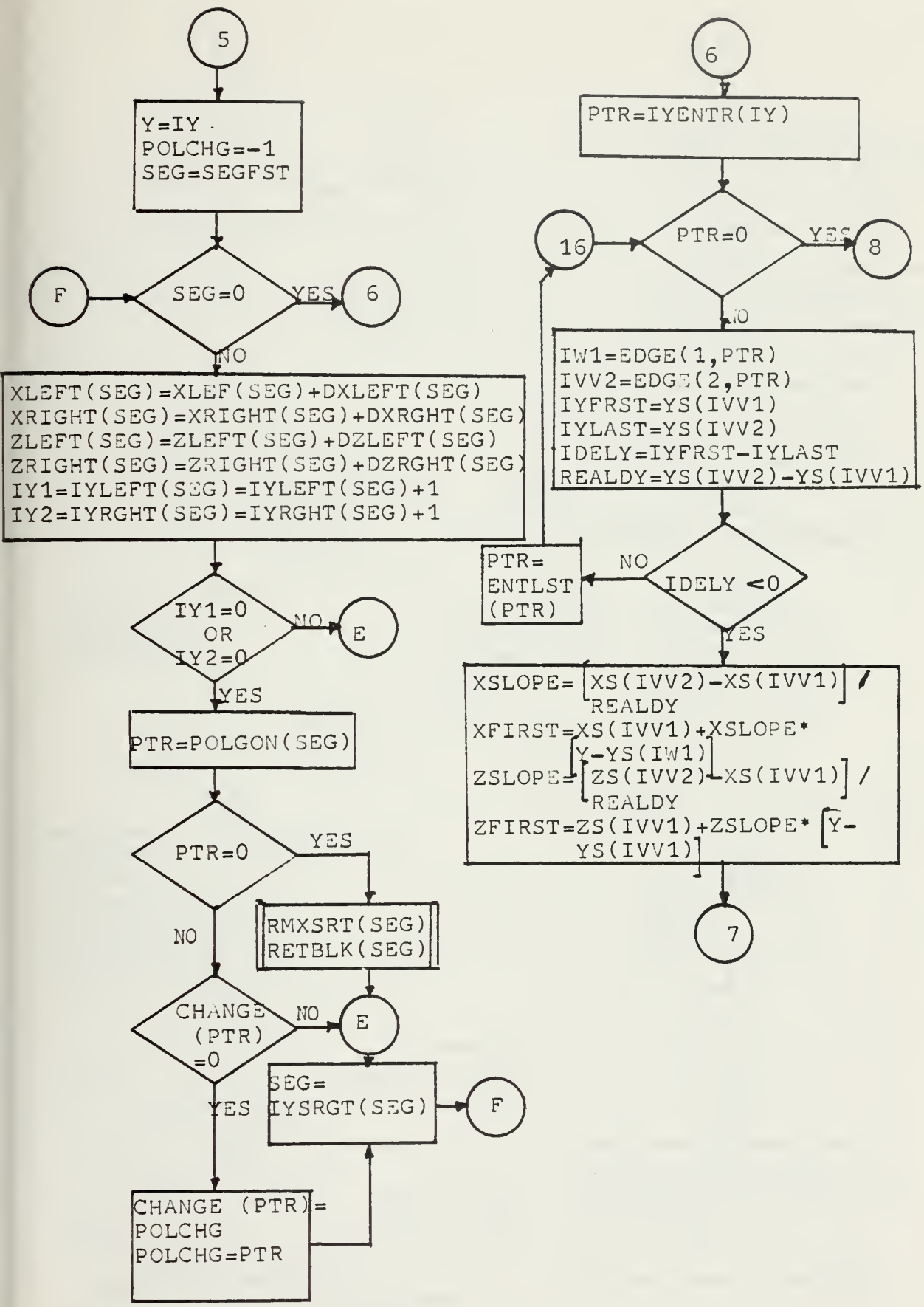
```
85      SEG=IXSRGT(SEG)
        GO TO 84
83      IF(CHG.EQ.1)GO TO 75
        SEGACT=0
        SEGCNT=0
87      IF(IMPLST.EQ.0) GO TO 86
        J=IMPLST
        IMPLST=IXSRGT(J)
        CALL RETBLK(J)
        GO TO 87
89      XSPNRG=XRESL
        GO TO 90
88      XSPNRG=XLSTUD
        GO TO 90
86      IMPLST=IMPLS2
        IMPLS2=0
        CURSEG=SEGFST
        XSPNRG=0.0
        SAMPLE=SAMFST
        SAMLST=0
        XLSTUD=0.0
98      XSPNLF=XSPNRG+1.0
        IF(XSPNLF.LE.XLSTUD) GO TO 88
        IF(SAMPLE.EQ.0) GO TO 89
        XSPNRG=SAMX(SAMPLE)
        IX=SAMPLE
        SAMPLE=SAMLNK(IX)
        SAMLNK(IX)=SAMFRE
        SAMFRE=IX
        XLSTUD=XSPNRG
90      IMPLFT=0
91      IBXCNT=0
        SEGOUT=0
        PREVIS=0
        SEG=SEGACT
92      IF(SEG.EQ.0) GO TO 94
        NEXT=IACTVF(SEG)
        XXX=XSPNRG+1.0
        IF(XRIGHT(SEG).GE.XXX) GO TO 95
        IF(PREVIS.EQ.0) GO TO 96
        IACTVE(PREVIS)=NEXT
97      IACTVE(SEG)=SEGOUT
        IF(SEGOUT.EQ.0) SEGLO=SEG
        SEGOUT=SEG
        IF(XRIGHT(SEG).GE.XSPNLF) GO TO (800,801),ILOOK
  800   CALL LOOKER(SEG)
        GO TO 802
  801   CALL LOOKR1(SEG)
  802   SEG=NEXT
        GO TO 92
96      SEGACT=NEXT
        GO TO 97
95      IF(XLEFT(SEG).LE.XSPNRG) GO TO (803,804),ILOOK
  803   CALL LOOKER(SEG)
        GO TO 805
  804   CALL LOOKR1(SEG)
  805   PREVIS=SEG
        SEG=NEXT
        GO TO 92
94      IF(CURSEG.EQ.0) GO TO 108
        SEG=CURSEG
        IF(XLEFT(SEG).GT.XSPNRG) GO TO 108
        CURSEG=IXSRGT(CURSEG)
```

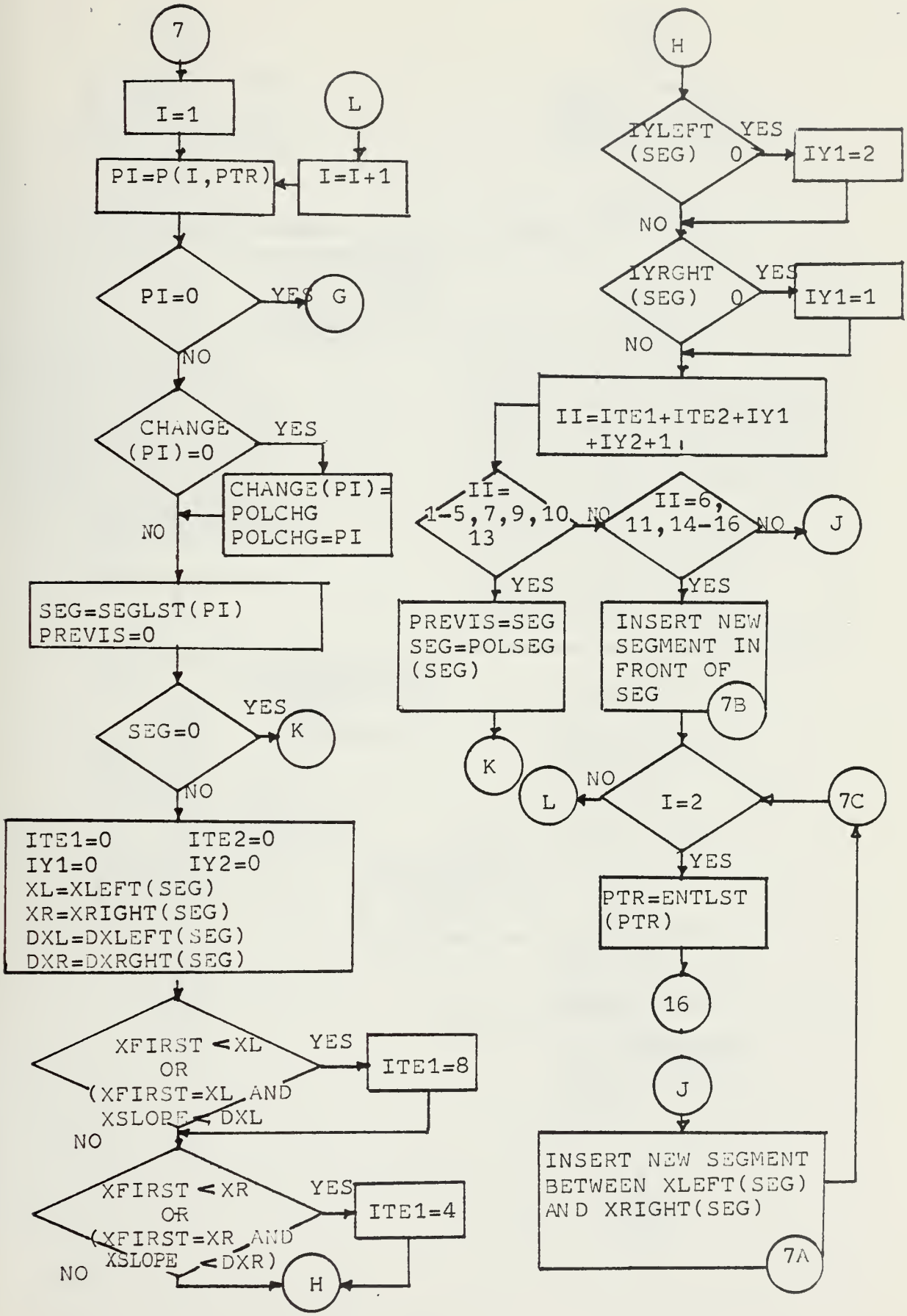
```
      IF(POLGON(SEG).NE.0) GO TO 99
      IF(XLEFT(SEG).LT.1.0) GO TO 600
      IF(XLEFT(SEG).GT.XRESL) GO TO 600
      IPO=POLSEG(SEG)/10000
      IF(IPO.NE.LSTSEG) GO TO 600
      IMPLFT=SEG
      GO TO 94
600    CALL RMXSRT(SEG)
      CALL RETBLK(SEG)
      GO TO 94
99    XXX=XLEFT(SEG)+1.0
      IF(XXX.GE.XRIGHT(SEG)) GO TO 94
      XSS=XSPNRG+1.0
      IF(XRIGHT(SEG).GE.XSS) GO TO 601
      IACTVE(SEG)=SEGOUT
      IF(SEGOUT.EQ.0) SEGLO=SEG
      SEGOUT=SEG
602    GO TO (806,807),ILOOK
 806   CALL LOOKER(SEG)
      GO TO 94
 807   CALL LOOKRT(SEG)
      GO TO 94
601    IACTVE(SEG)=SEGACT
      SEGACT=SEG
      GO TO 602
108   CALL THINKR(ITR,SEG)
      GO TO (603,160),ITR
      IF(SEGOUT.EQ.0) GO TO 604
      IACTVE(SEGLO)=SEGACT
      SEGACT=SEGOUT
604    I=DIV
      ISSRGT=XSPNRG
      IF(I.LT.ISSRGT) GO TO 605
      I=(XSPNLF+XSPNRG)/2.0
605    XSPNRG=I
      GO TO 91
603    IF(IMPLFT.EQ.0) GO TO 606
      CALL RMXSRT(IMPLFT)
      CALL RETBLK(IMPLFT)
606    IF(XSPNRG.LT.XRESL) GO TO 98
      IF(SAMLST.EQ.0) GO TO 610
      SAMLNK(SAMLST)=0
      GO TO 611
610    SAMFST=0
611   CALL SHOW
59    CONTINUE
      RETURN
100   WRITE(6,121)
121   FORMAT('ERROR: THE Y COORDINATE IS OFF THE SCREEN.')
      RETURN
110   WRITE(6,111)
111   FORMAT('ERROR: NEXT=0')
      RETURN
 160  WRITE(6,999)
 999  FORMAT(2X,'THE NUMBER OF SEGMENTS GENERATED FOR THIS SCAN
     &LINE',/2X,'THE STORAGE PROVIDED.')
      RETURN
      END
```

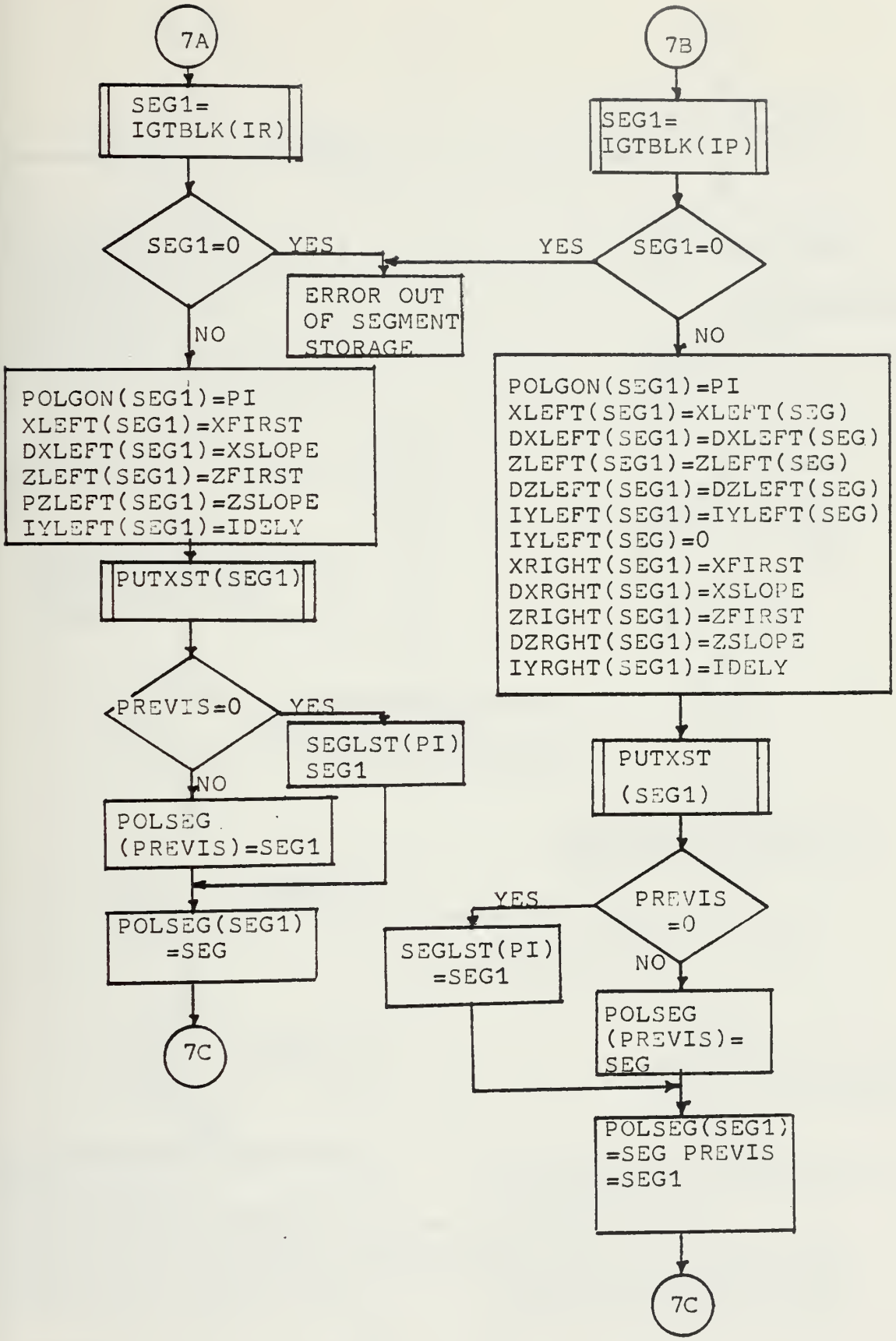

SURFAC (ISR,ILOOK)
CONVERT OBJECT COORDINATES TO SCREEN COORDINATES
1
ZERO ARRAY IYFNTR(), LINK ARRAYS INACTIVE() AND SAMLNK ON THEMSELVES
2
INITIALIZE POINTERS
3
EXAMINE EDGES AND DETERMINE SCAN LINE EACH ENTERS THE DISPLAY. STORE THIS INFO. IN IYENTR() AND ENTLST()
4
IY=1 -IY IS THE SCAN LINE POINTER
C
UPDATE SEGMENT ENDPOINTS USING SLOPES OF THE X & Z 2-D LINE EQN.'S AND MARK THE POLYGON OF AN EXITING SEGMENT AS CHANGING FOR ALL ACTIVE SEGMENTS
5
B
COMPUTE SLOPES OF THE X & Z 2-D LINES AND THE INTER-SECTION OF ALL ENTERING EDGES WITH SCAN LINE IY
6
PLACE EACH ENTERING EDGE INTO THE PROPER LOCATION OF THE TWO POLYGONS SEGMENT LISTS
7
REMOVE ALL EXITING EDGES FROM POLYGONAL SEGMENT LISTS, AND CONSOLIDATE SEGMENT BLOCK STORAGE
8
A
A
ORDER ACTIVE LIST OF SEGMENTS BY INCREASING X VALUE OF THE LEFT ENDPOINT
9
SEGCNT=SEGACT=0 AND REMOVE ALL EXITING IMPLIED EDGES
10
INITIALIZE POINTS AND X VALUE OF FIRST SPAN (XSPNRG) TO ZERO. USING PREVIS SCAN LINES SAMPLE POINTS DIVIDE SCAN LINE IY INTO SPANS.
11
B
PUT ALL SEGMENTS WHICH INTERSECT THIS SPAN INTO TWO LISTS, SEGOUT AND SEGACT.
12
HAVE LOOKER OR LOOKR1 EXAMINE ALL SEGMENTS IN THESE TWO LISTS TO DETERMINE BOX COUNT AND BOX TYPE
13
FAILURE
HAVE THINKER DETERMINE IF THE CONTENTS OF THIS SPAN CAN BE DISPLAYED
14
SUCCESS
XSPNRG= XRESL
NO
B
SHOW SCAN LINE IY
C
IY=IY+1
IY=IYRES
YES
RETURN

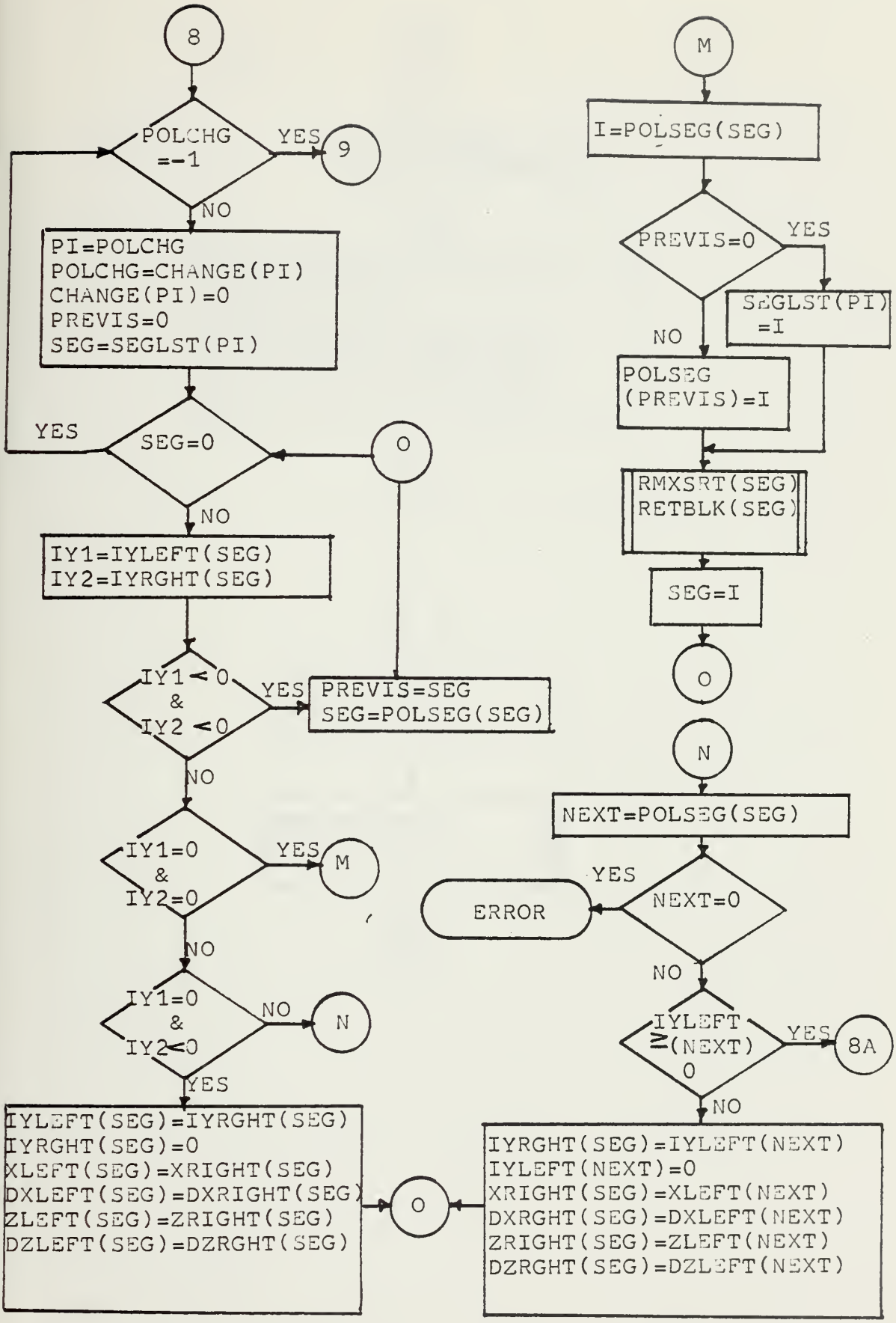

1
RDYCLP
CLIP
SCRN
INIT2(ISR)
IYRES=2*VSY
IXRESL=XRESL=2*VSX
SHOWIN
(IYRES)
2
ZERO ARRAY
IYENTR()
MAXSEG=60 M=MAXSEG
MM=M-1
IACTIVE(M)=0
D O
I=1,MM
IACTIVE(I)=I+1
M=MAXSEG*2 MM=M-1
SAMLNK(M)=0
D O
I=1,MM
SAMLNK(I)=
I+1
3
IFRELS=1
SAMFRE=1
IMPLST=0
IMPLS2=0
SEGFST=0
PTR=EDGLST

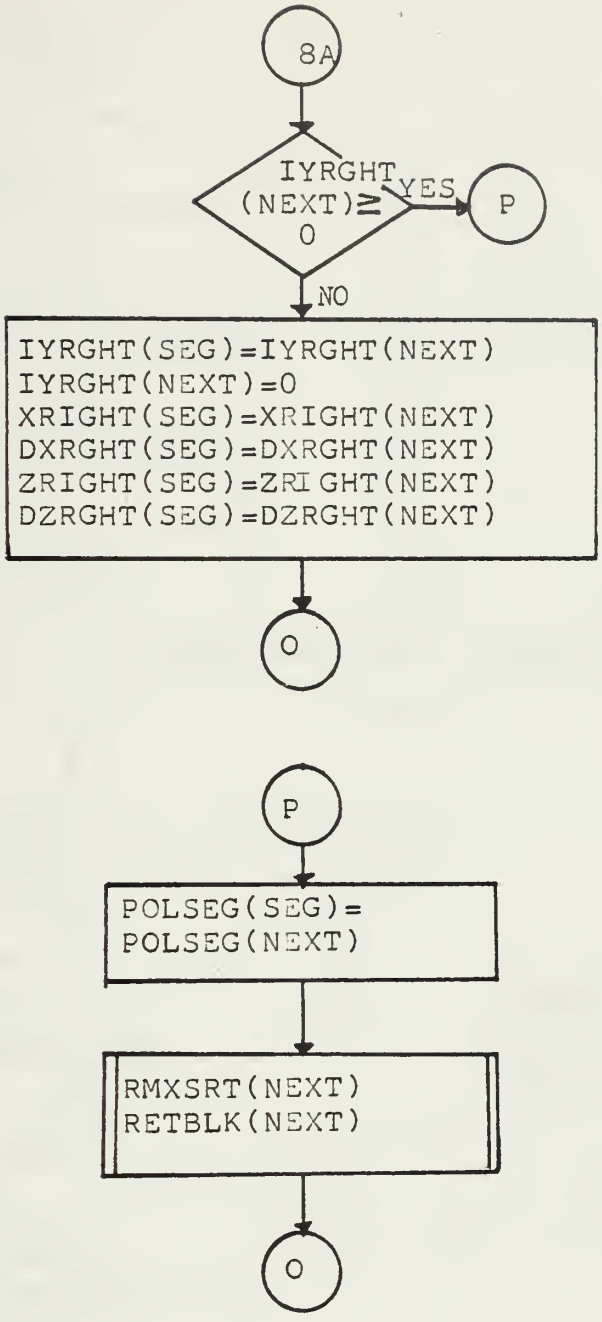

4
D
PTR=0
YES
5
NO
NEXT=ENTLST(PTR)
I=P(1,PTR)
I=P(2,PTR)
NO
I=0
AND
SHAD(I)=0
I=0
AND
SHAD(I)=0
YES
YES
NO
PTR=NEXT
D
J=EDGE(1,PTR)
K=EDGE(2,PTR)
YS(J)>YS(K)
YES
ISWAP
EDGE(1,PTR),
EDGE(2,PTR)
NO
J=K
I=YS(J)+0,99999
1 ≤I≤
IYRESL
NO
ERROR
YES
ENTLST(PTR)=IYENTR(I)
IYENTR(I)=PTR
PTR=NEXT
D

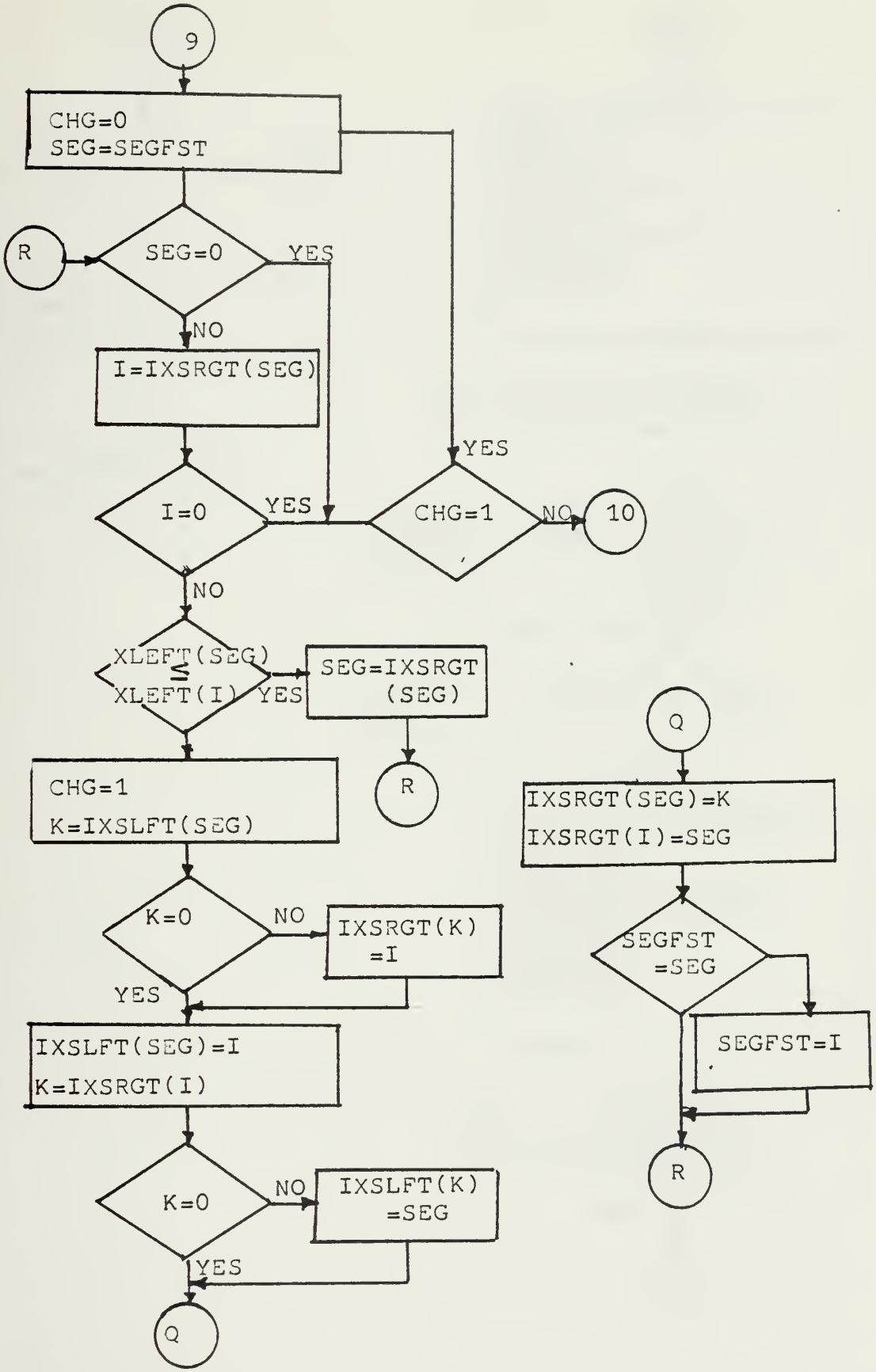

5
Y=IY
POLCHG=-1
SEG=SEGFST
F
SEG=0
YES
6
NO
XLEFT(SEG)=XLEF(SEG)+DXLEFT(SEG)
XRIGHT(SEG)=XRIGHT(SEG)+DXRGHT(SEG)
ZLEFT(SEG)=ZLEFT(SEG)+DZLEFT(SEG)
ZRIGHT(SEG)=ZRIGHT(SEG)+DZRGHT(SEG)
IY1=IYLEFT(SEG)=IYLEFT(SEG)+1
IY2=IYRGHT(SEG)=IYRGHT(SEG)+1
IY1=0 OR IY2=0
NO
E
YES
PTR=POLGON(SEG)
PTR=0
YES
RMXSRT(SEG)
RETBLK(SEG)
NO
CHANGE (PTR) =0
NO
E
YES
SEG= IYSRGT(SEG)
F
CHANGE (PTR)= POLCHG
POLCHG=PTR
6
PTR=IYENTR(IY)
16
PTR=0
YES
8
NO
IW1=EDGE(1,PTR)
IVV2=EDGE(2,PTR)
IYFRST=YS(IVV1)
IYLAST=YS(IVV2)
IDELY=IYFRST-IYLAST
REALDY=YS(IVV2)-YS(IVV1)
IDELY <0
NO
PTR= ENTLST (PTR)
YES
XSLOPE= [XS(IVV2)-XS(IVV1)] / REALDY
XFIRST=XS(IVV1)+XSLOPE* [Y-YS(IW1)]
ZSLOPE= [ZS(IVV2)-XS(IVV1)] / REALDY
ZFIRST=ZS(IVV1)+ZSLOPE* [Y-YS(IVV1)]
7

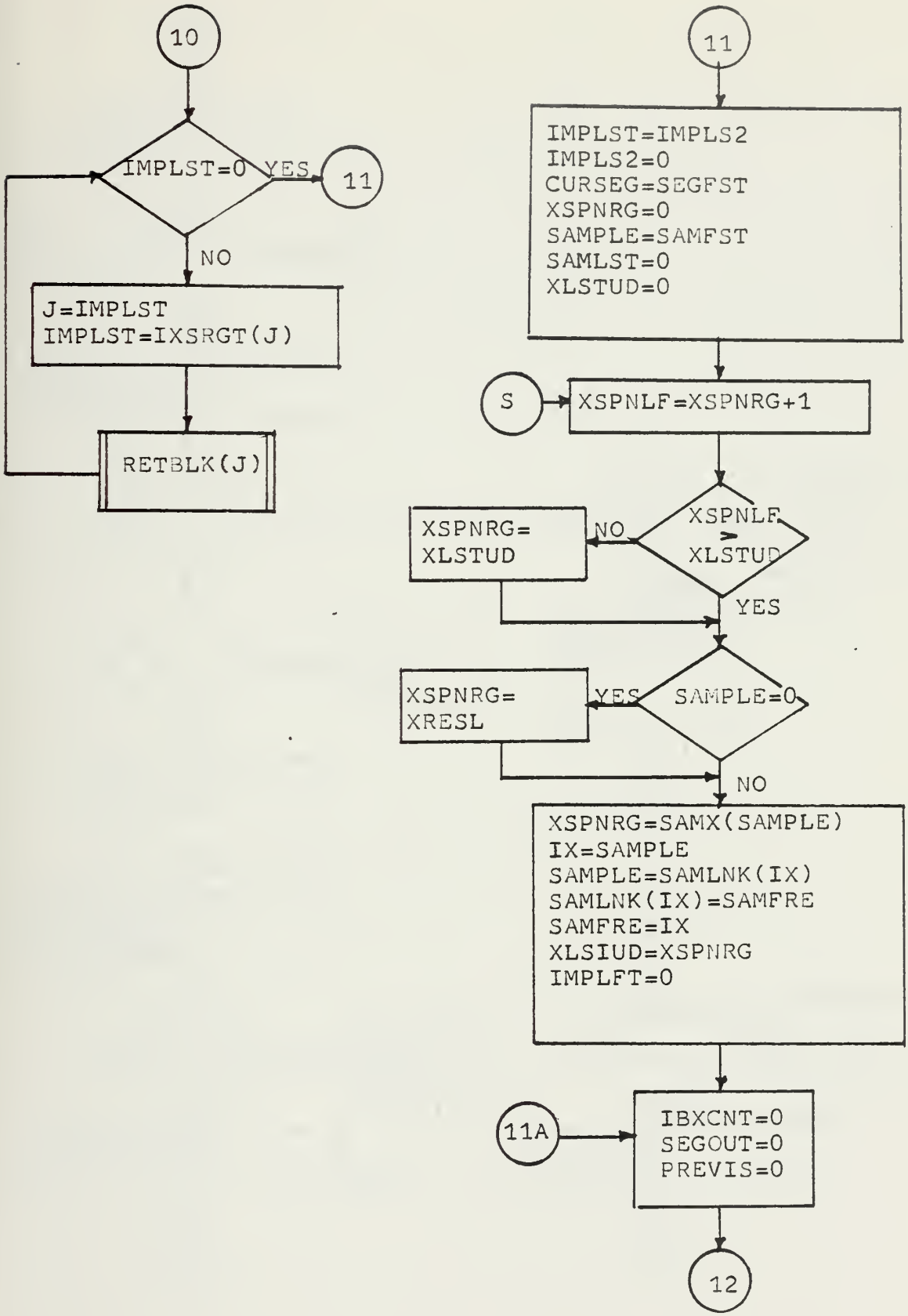

7
I=1
L
PI=P(I,PTR)
I=I+1
PI=0
YES
G
NO
CHANGE (PI)=0
YES
CHANGE(PI)=POLCHG
POLCHG=PI
NO
SEG=SEGLST(PI)
PREVIS=0
SEG=0
YES
K
NO
ITE1=0 ITE2=0
IY1=0 IY2=0
XL=XLEFT(SEG)
XR=XRIGHT(SEG)
DXL=DXLEFT(SEG)
DXR=DXRGHT(SEG)
XFIRST <XL OR (XFIRST=XL AND XSLOPE <DXL
YES
ITE1=8
NO
XFIRST <XR OR (XFIRST=XR AND XSLOPE <DXR)
YES
ITE1=4
NO
H
H
IYLEFT (SEG) 0
YES
IY1=2
NO
IYRGHT (SEG) 0
YES
IY1=1
NO
II=ITE1+ITE2+IY1+IY2+1
II=1-5,7,9,10,13
NO
II=6,11,14-16
NO
J
YES
YES
PREVIS=SEG
SEG=POLSEG(SEG)
INSERT NEW SEGMENT IN FRONT OF SEG
7B
K
L
NO
I=2
7C
YES
PTR=ENTLST(PTR)
16
J
INSERT NEW SEGMENT BETWEEN XLEFT(SEG) AND XRIGHT(SEG)
7A

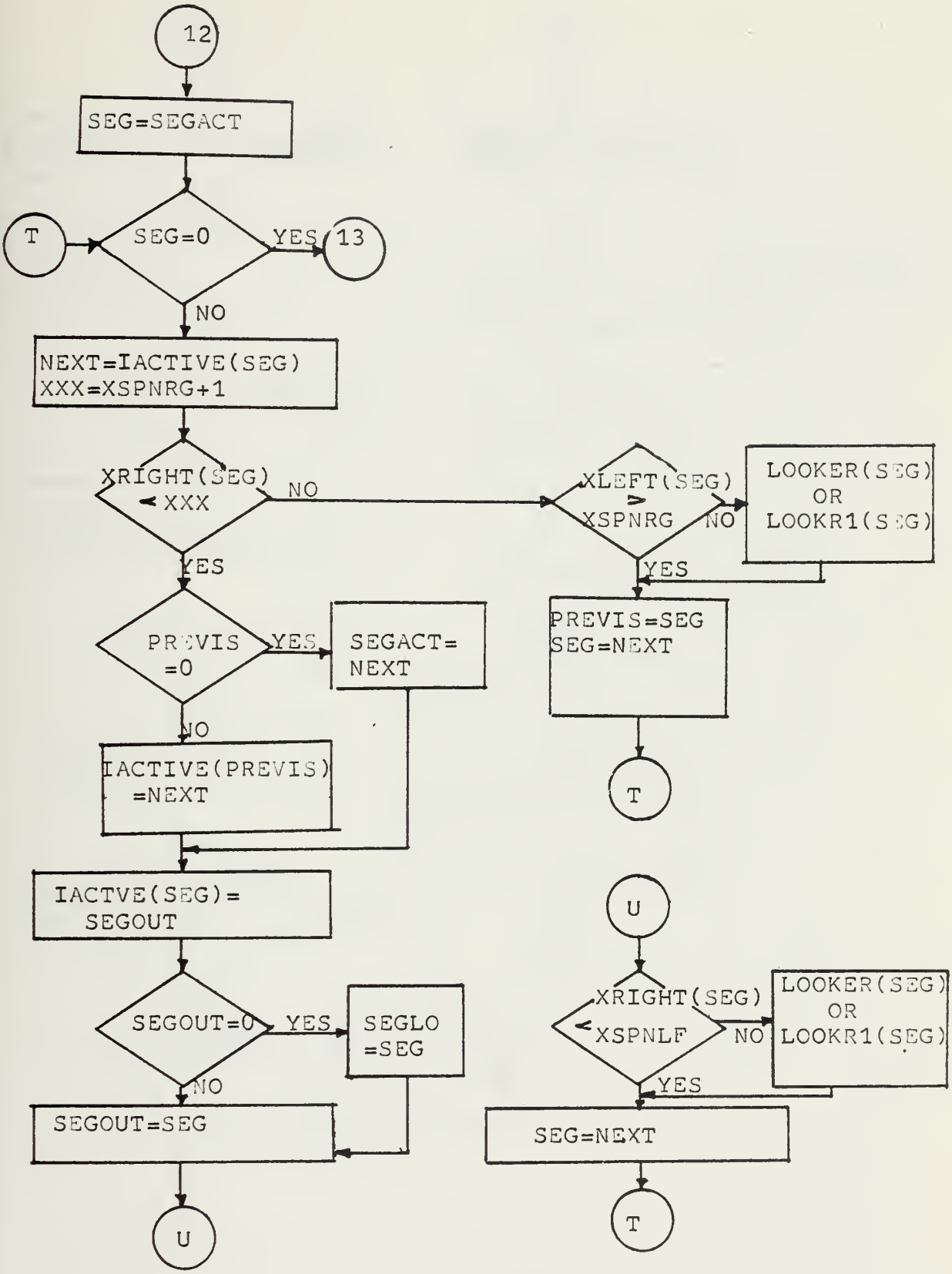

7A
SEG1=
IGTBLK(IR)
SEG1=0
YES
NO
ERROR OUT
OF SEGMENT
STORAGE
POLGON(SEG1)=PI
XLEFT(SEG1)=XFIRST
DXLEFT(SEG1)=XSLOPE
ZLEFT(SEG1)=ZFIRST
PZLEFT(SEG1)=ZSLOPE
IYLEFT(SEG1)=IDELY
PUTXST(SEG1)
PREVIS=0
YES
NO
SEGLST(PI)
SEG1
POLSEG
(PREVIS)=SEG1
POLSEG(SEG1)
=SEG
7C
7B
SEG1=
IGTBLK(IP)
YES
SEG1=0
NO
POLGON(SEG1)=PI
XLEFT(SEG1)=XLEFT(SEG)
DXLEFT(SEG1)=DXLEFT(SEG)
ZLEFT(SEG1)=ZLEFT(SEG)
DZLEFT(SEG1)=DZLEFT(SEG)
IYLEFT(SEG1)=IYLEFT(SEG)
IYLEFT(SEG)=0
XRIGHT(SEG1)=XFIRST
DXRGHT(SEG1)=XSLOPE
ZRIGHT(SEG1)=ZFIRST
DZRGHT(SEG1)=ZSLOPE
IYRGHT(SEG1)=IDELY
PUTXST
(SEG1)
PREVIS
=0
YES
NO
SEGLST(PI)
=SEG1
POLSEG
(PREVIS)=
SEG
POLSEG(SEG1)
=SEG PREVIS
=SEG1
7C

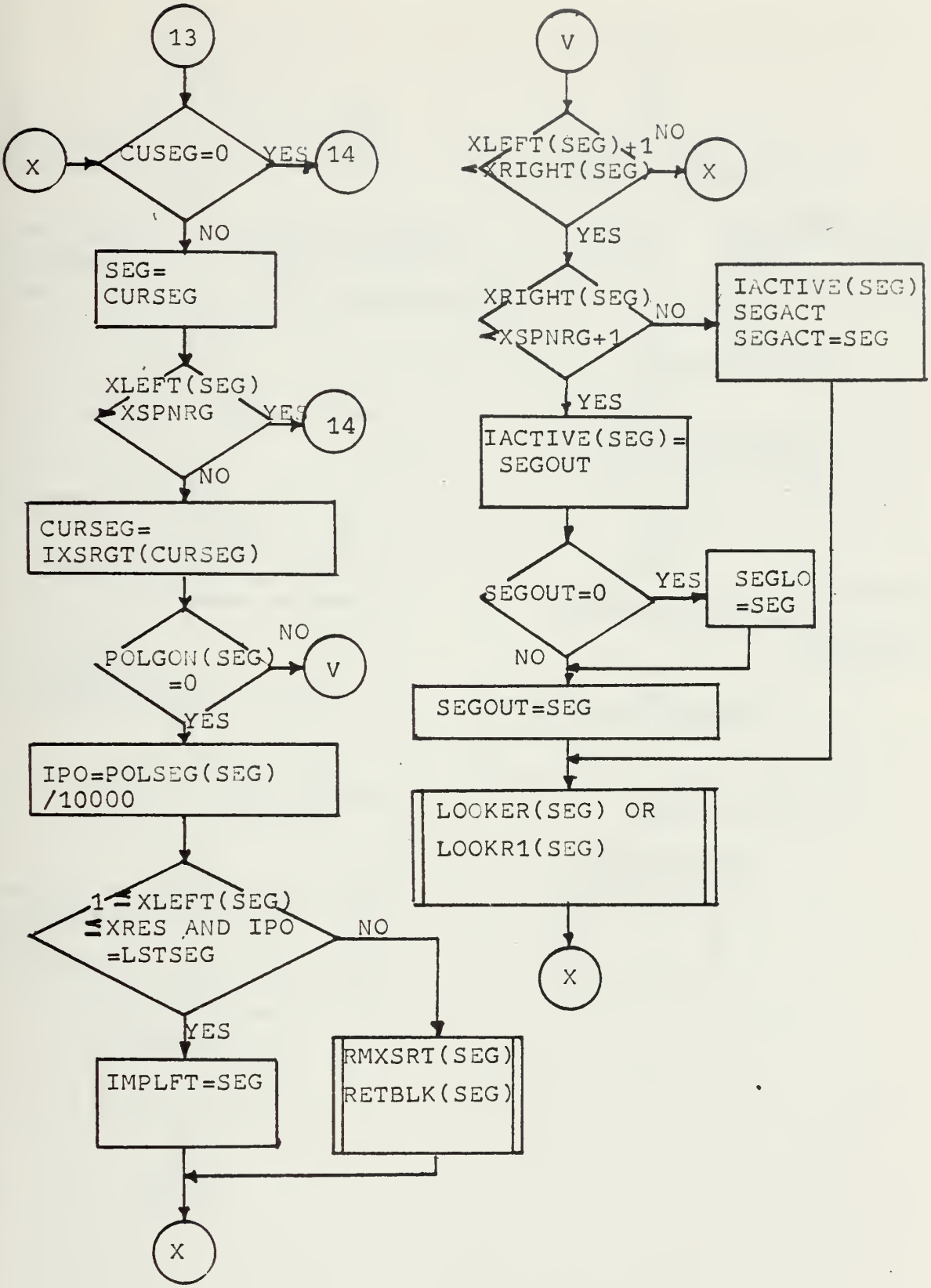

8
POLCHG =-1
YES
9
NO
PI=POLCHG
POLCHG=CHANGE(PI)
CHANGE(PI)=0
PREVIS=0
SEG=SEGLST(PI)
YES
SEG=0
O
NO
IY1=IYLEFT(SEG)
IY2=IYRGHT(SEG)
IY1<0 & IY2<0
YES
PREVIS=SEG
SEG=POLSEG(SEG)
NO
IY1=0 & IY2=0
YES
M
NO
IY1=0 & IY2<0
NO
N
YES
IYLEFT(SEG)=IYRGHT(SEG)
IYRGHT(SEG)=0
XLEFT(SEG)=XRIGHT(SEG)
DXLEFT(SEG)=DXRIGHT(SEG)
ZLEFT(SEG)=ZRIGHT(SEG)
DZLEFT(SEG)=DZRGHT(SEG)
O
M
I=POLSEG(SEG)
PREVIS=0
YES
SEGLST(PI) =I
NO
POLSEG (PREVIS)=I
RMXSRT(SEG)
RETBLK(SEG)
SEG=I
O
N
NEXT=POLSEG(SEG)
YES
NEXT=0
ERROR
NO
IYLEFT (NEXT) ≥ 0
YES
8A
NO
IYRGHT(SEG)=IYLEFT(NEXT)
IYLEFT(NEXT)=0
XRIGHT(SEG)=XLEFT(NEXT)
DXRGHT(SEG)=DXLEFT(NEXT)
ZRIGHT(SEG)=ZLEFT(NEXT)
DZRGHT(SEG)=DZLEFT(NEXT)

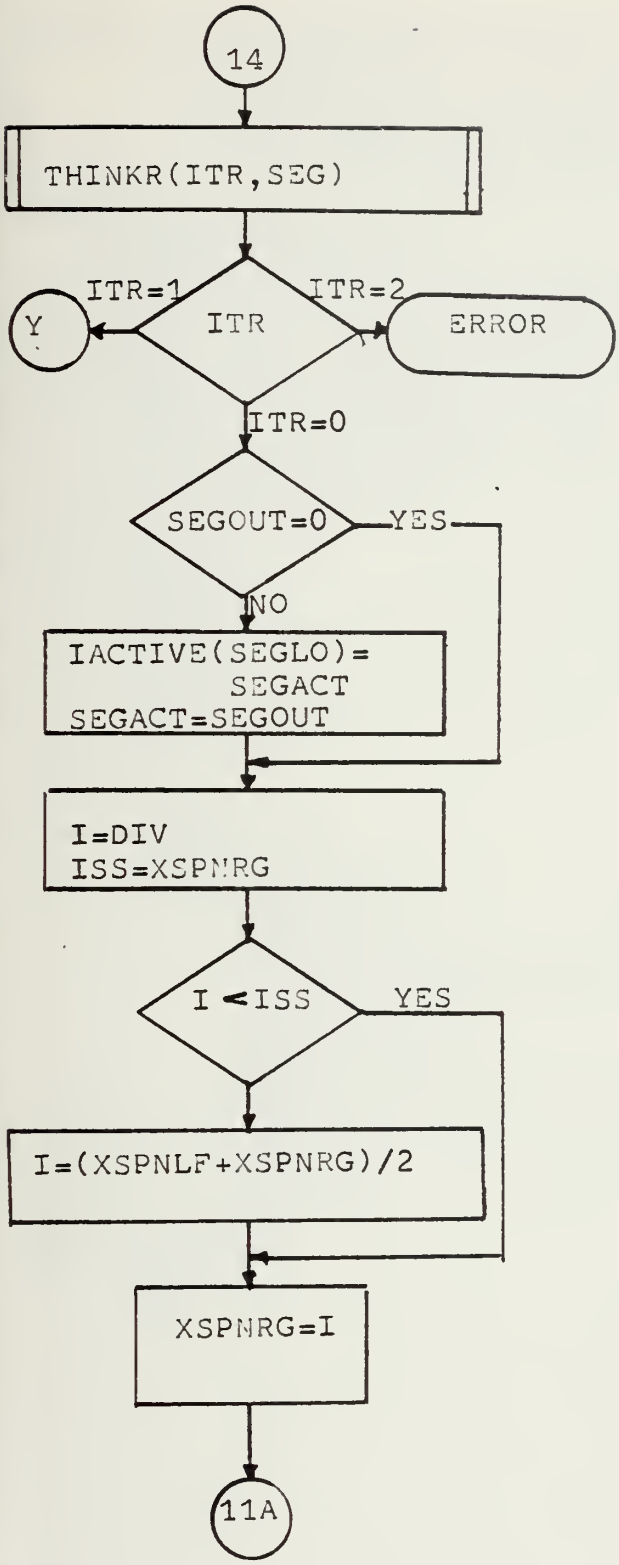

8A
IYRGHT (NEXT) ≥ 0
YES
P
NO
IYRGHT(SEG)=IYRGHT(NEXT)
IYRGHT(NEXT)=0
XRIGHT(SEG)=XRIGHT(NEXT)
DXRGHT(SEG)=DXRGHT(NEXT)
ZRIGHT(SEG)=ZRIGHT(NEXT)
DZRGHT(SEG)=DZRGHT(NEXT)
O
P
POLSEG(SEG)=
POLSEG(NEXT)
RMXSRT(NEXT)
RETBLK(NEXT)
O

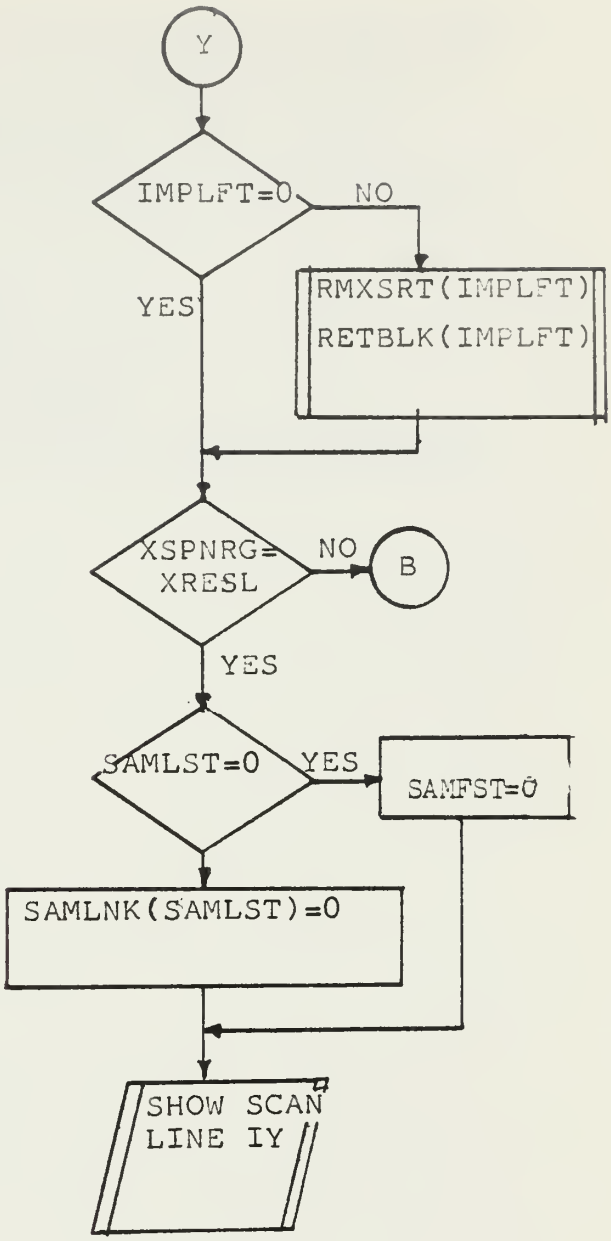

9
CHG=0
SEG=SEGFST
R
SEG=0
YES
NO
I=IXSRGT(SEG)
YES
I=0
YES
CHG=1
NO
10
NO
XLEFT(SEG)
≤
XLEFT(I)
YES
SEG=IXSRGT
(SEG)
R
CHG=1
K=IXSLFT(SEG)
K=0
NO
IXSRGT(K)
=I
YES
IXSLFT(SEG)=I
K=IXSRGT(I)
K=0
NO
IXSLFT(K)
=SEG
YES
Q
Q
IXSRGT(SEG)=K
IXSRGT(I)=SEG
SEGFST
=SEG
SEGFST=I
R


10
IMPLST=0
YES
11
NO
J=IMPLST
IMPLST=IXSRGT(J)
RETBLK(J)
11
IMPLST=IMPLS2
IMPLS2=0
CURSEG=SEGFST
XSPNRG=0
SAMPLE=SAMFST
SAMLST=0
XLSTUD=0
S
XSPNLF=XSPNRG+1
XSPNLF
>
XLSTUD
NO
XSPNRG=
XLSTUD
YES
SAMPLE=0
YES
XSPNRG=
XRESL
NO
XSPNRG=SAMX(SAMPLE)
IX=SAMPLE
SAMPLE=SAMLNK(IX)
SAMLNK(IX)=SAMFRE
SAMFRE=IX
XLSIUD=XSPNRG
IMPLFT=0
11A
IBXCNT=0
SEGOUT=0
PREVIS=0
12

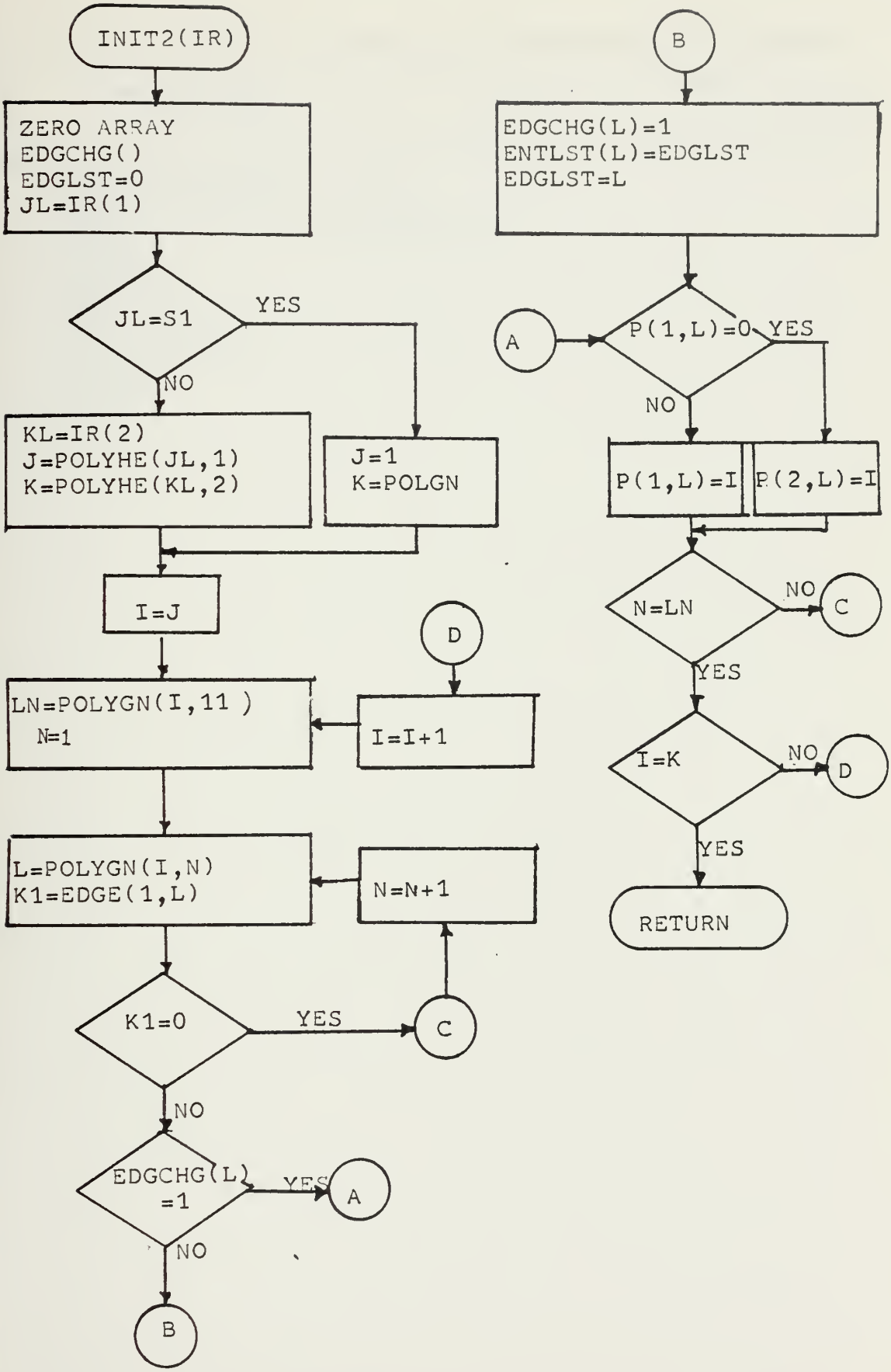

12
SEG=SEGACT
T
SEG=0
YES
13
NO
NEXT=IACTIVE(SEG)
XXX=XSPNRG+1
XRIGHT(SEG)
< XXX
NO
XLEFT(SEG)
>
XSPNRG
NO
LOOKER(SEG)
OR
LOOKR1(SEG)
YES
YES
PREVIS
=0
YES
SEGACT=
NEXT
PREVIS=SEG
SEG=NEXT
NO
IACTIVE(PREVIS)
=NEXT
T
IACTVE(SEG)=
SEGOUT
U
SEGOUT=0
YES
SEGLO
=SEG
XRIGHT(SEG)
< XSPNLF
NO
LOOKER(SEG)
OR
LOOKR1(SEG)
NO
YES
SEGOUT=SEG
SEG=NEXT
U
T


13
X
CUSEG=0
YES
14
NO
SEG=
CURSEG
XLEFT(SEG)
XSPNRG
YES
14
NO
CURSEG=
IXSRGT(CURSEG)
POLGON(SEG)
=0
NO
V
YES
IPO=POLSEG(SEG)
/10000
1=XLEFT(SEG)
=XRES AND IPO
=LSTSEG
NO
YES
IMPLFT=SEG
RMXSRT(SEG)
RETBLK(SEG)
X
V
XLEFT(SEG)+1
XRIGHT(SEG)
NO
X
YES
XRIGHT(SEG)
XSPNRG+1
NO
IACTIVE(SEG)
SEGACT
SEGACT=SEG
YES
IACTIVE(SEG)=
SEGOUT
SEGOUT=0
YES
SEGLO
=SEG
NO
SEGOUT=SEG
LOOKER(SEG) OR
LOOKR1(SEG)
X

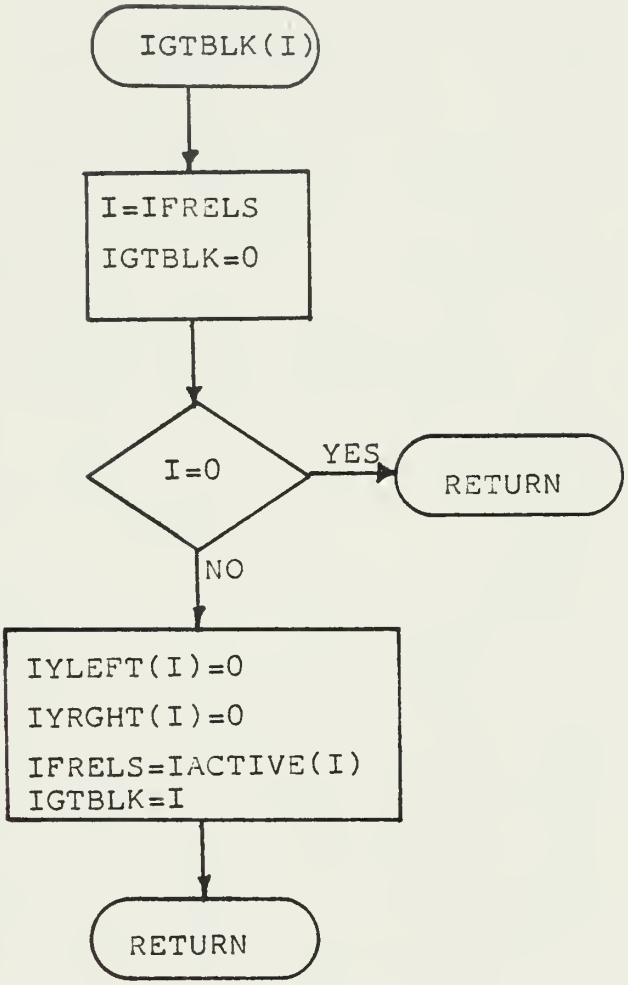

14
THINKR(ITR,SEG)
ITR
ITR=1
ITR=2
Y
ERROR
ITR=0
SEGOUT=0
YES
NO
IACTIVE(SEGLO)=
SEGACT
SEGACT=SEGOUT
I=DIV
ISS=XSPNRG
I <ISS
YES
I=(XSPNLF+XSPNRG)/2
XSPNRG=I
11A


Y
IMPLFT=0
NO
YES
RMXSRT(IMPLFT)
RETBLK(IMPLFT)
XSPNRG=
XRESL
NO
B
YES
SAMLST=0
YES
SAMFST=0
SAMLNK(SAMLST)=0
SHOW SCAN
LINE IY

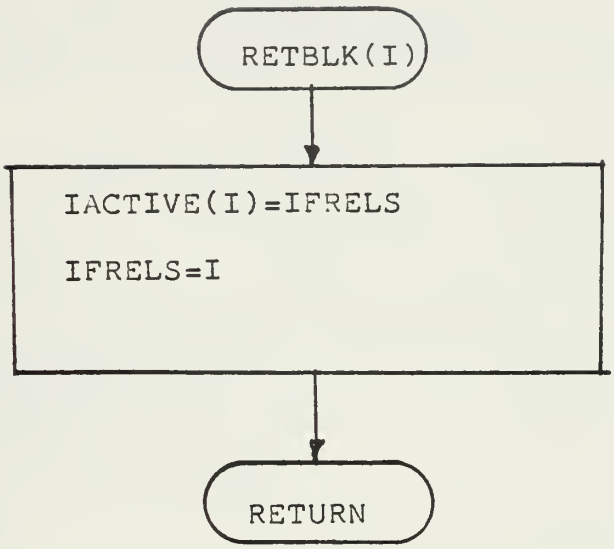
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       INIT2: LINKS THE LIST OF EDGES IN THE ARRAY ENTLST(I) AND
C           STORES THE INDEX OF THE POLYGONS WHICH HAVE EDGE I
C           AS A BOUNDRY IN THE ARRAY P(2,I).
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE INIT2(IR)
      DIMENSION IR(2)
      COMMON /AA/ POLYHE(10,2),POLYHN
      COMMON /AB/ POLYGN(60,11),POLGN,SHAD(60)
      COMMON /AAD/ EDGE(2,200),EDGEM
      COMMON /AAB/ XS(120),YS(120),ZS(120),POINTM
      COMMON /BA/ENTLST(200),P(2,200),EDGLST
      COMMON /FF/EDGCHG(200)
      INTEGER EDGE,EDGEM,POLYGN,POLGN,POINTM,SHAD,ENTLST,P,EDGLST,
     &POLYHE,POLYHN,EDGCHG
      DO 30 I=1,EDGEM
   30   EDGCHG(I)=0
      EDGLST=0
      JL=IR(1)
      IF(JL.EQ.31)GO TO 1000
      KL=IR(2)
      J=POLYHE(JL,1)
      K=POLYHE(KL,2)
      GO TO 500
 1000 J=1
      K=POLGN
  500 DO 1010 I=J,K
        LN=POLYGN(I,11)
        DO 1020 N=1,LN
            L=POLYGN(I,N)
            K1=EDGE(1,L)
            IF(K1.EQ.0) GO TO 1020
            IF(EDGCHG(L).EQ.1) GO TO 520
            EDGCHG(L)=1
            ENTLST(L)=EDGLST
            EDGLST=L
  520       IF(P(1,L).EQ.0) GO TO 521
            P(2,L)=I
            GO TO 1020
  521       P(1,L)=I
 1020   CONTINUE
 1010 CONTINUE
      RETURN
      END
```


INIT2(IR)
ZERO ARRAY
EDGCHG()
EDGLST=0
JL=IR(1)
JL=S1
YES
NO
KL=IR(2)
J=POLYHE(JL,1)
K=POLYHE(KL,2)
J=1
K=POLGN
I=J
D
LN=POLYGN(I,11)
N=1
I=I+1
L=POLYGN(I,N)
K1=EDGE(1,L)
N=N+1
K1=0
YES
C
NO
EDGCHG(L)
=1
YES
A
NO
B
B
EDGCHG(L)=1
ENTLST(L)=EDGLST
EDGLST=L
A
P(1,L)=0
YES
NO
P(1,L)=I
P(2,L)=I
N=LN
NO
C
YES
I=K
NO
D
YES
RETURN

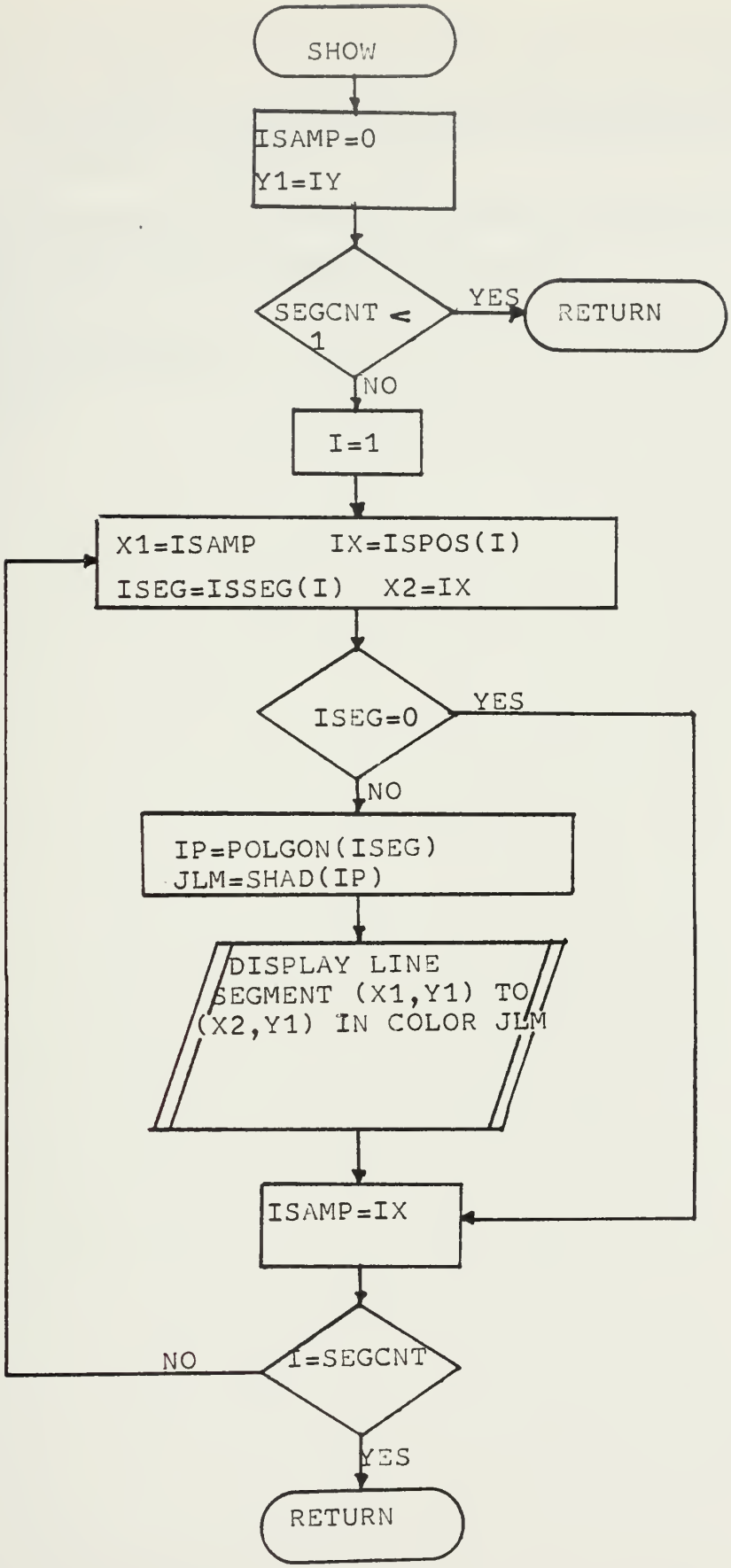
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C        IGTBLK: IS USED TO OBTAIN THE INDEX OF A FREE SEGMENT BLOCK
C            OF STORAGE.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      FUNCTION IGTBLK(I)
      COMMON /IA/IACTVE(60),IFRELS
      COMMON /IB/ IYLEFT(60),IYRGHT(60)
      I=IFRELS
      IGTBLK=0
      IF(I.EQ.0) RETURN
      IYLEFT(I)=0
      IYRGHT(I)=0
      IFRELS=IACTVE(I)
      IGTBLK=I
      RETURN
      END
```


IGTBLK(I)
I=IFRELS
IGTBLK=0
I=0
YES
RETURN
NO
IYLEFT(I)=0
IYRGHT(I)=0
IFRELS=IACTIVE(I)
IGTBLK=I
RETURN

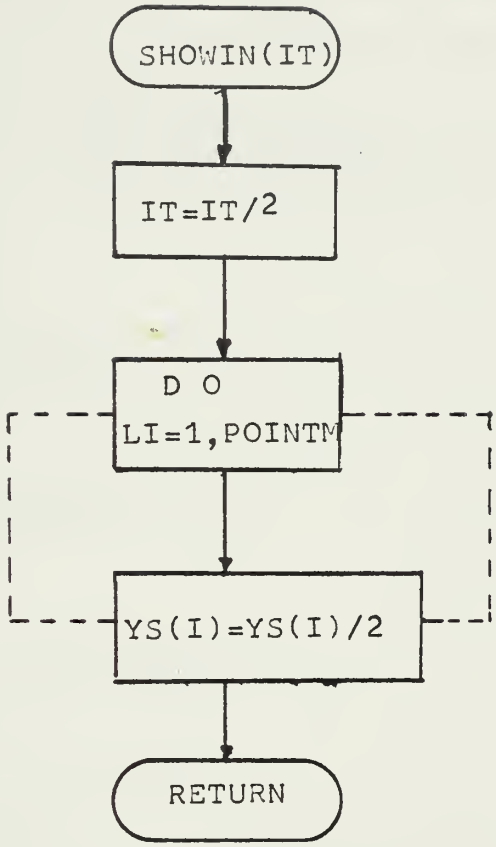
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       RETBLK: RETURNS A SEGMENT'S BLOCK OF STORAGE TO THE FREE
C           LIST.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE RETBLK(I)
      COMMON /TA/IACTVE(60),IFRELS
      IACTVE(I)=IFRELS
      IFRELS=I
      RETURN
      END
```


RETBLK(I)
IACTIVE(I)=IFRELS
IFRELS=I
RETURN

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C        SHOW: DISPLAYS EACH SEGMENT WITH THE APPROPRIATE POLYGONAL
C              COLOR OR SHADE ON THE RAMTEK SCREEN.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE SHOW
      COMMON /TO/ SEGCNT,LSTSEG,IY
      COMMON /TK/ POLSEG(60),POLGON(60)
      COMMON /AB/ POLYGN(60,11),POLGN,SHAD(60)
      COMMON /TP/ISPOS(60),ISSEG(60)
      INTEGER SEGCNT,POLSEG,POLGON,POLYGN,POLGN,SHAD,VECTOR
      INTEGER COLOR
      ISAMP=0
      Y1=IY
      IF(SEGCNT.LT.1) RETURN
      DO 700 I=1,SEGCNT
         X1=ISAMP
         ISEG=ISSEG(I)
         IX=ISPOS(I)
         X2=IX
         IF(ISEG.EQ.0) GO TO 710
         IP=POLGON(ISEG)
         JLM=SHAD(IP)
         II=COLOR(JLM)
         KL=VECTOR(X1,Y1,X2,Y1)
         IF(KL.LT.0) GO TO 1554
  710    ISAMP=IX
  700  CONTINUE
       RETURN
 1554  WRITE(6,1)
    1  FORMAT(2X,'THE FUNCTION VECTOR FAILED')
       RETURN
       END
```

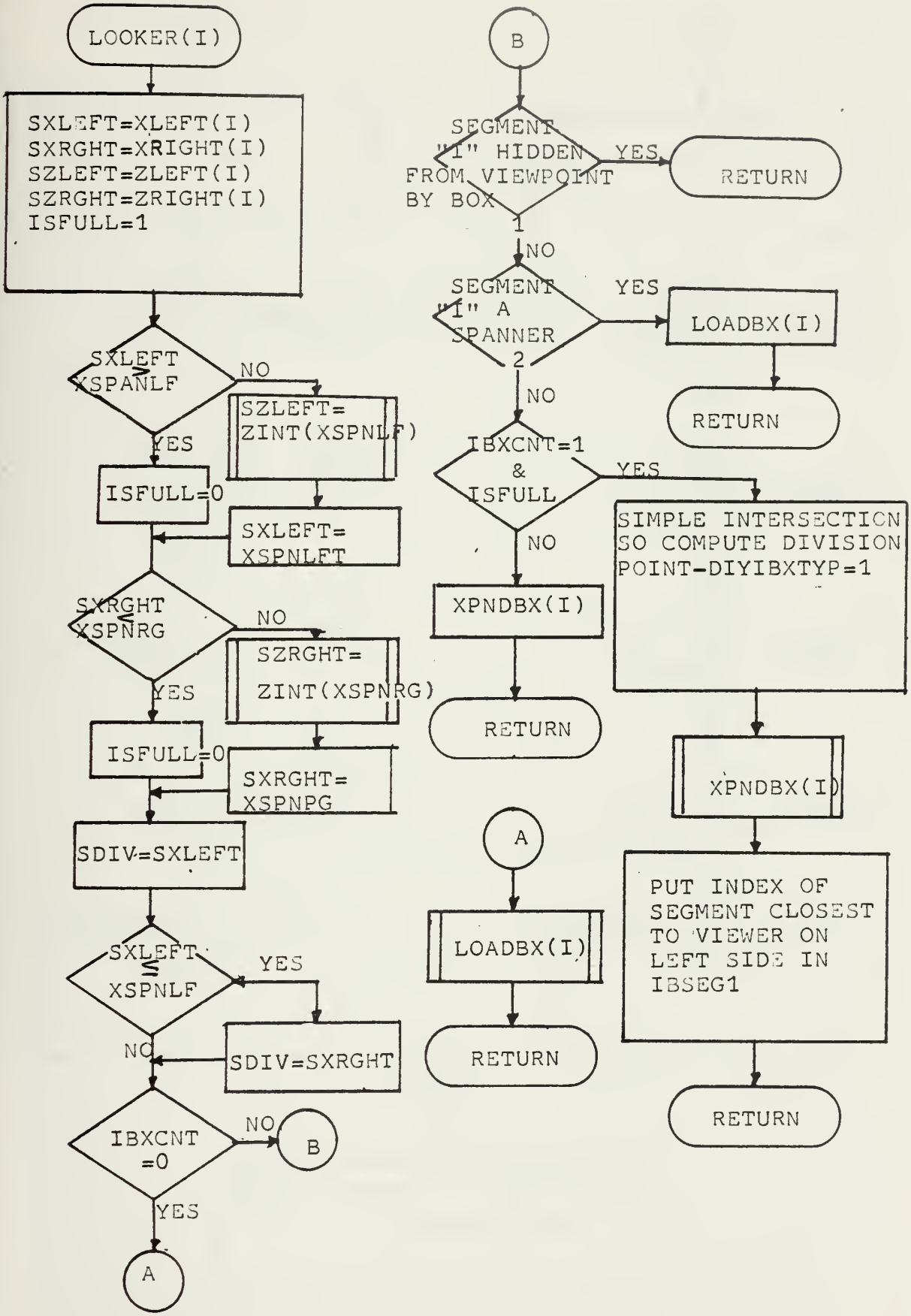

SHOW
ISAMP=0
Y1=IY
SEGCNT < 1
YES
RETURN
NO
I=1
X1=ISAMP IX=ISPOS(I)
ISEG=ISSEG(I) X2=IX
ISEG=0
YES
NO
IP=POLGON(ISEG)
JLM=SHAD(IP)
DISPLAY LINE SEGMENT (X1,Y1) TO (X2,Y1) IN COLOR JLM
ISAMP=IX
I=SEGCNT
NO
YES
RETURN

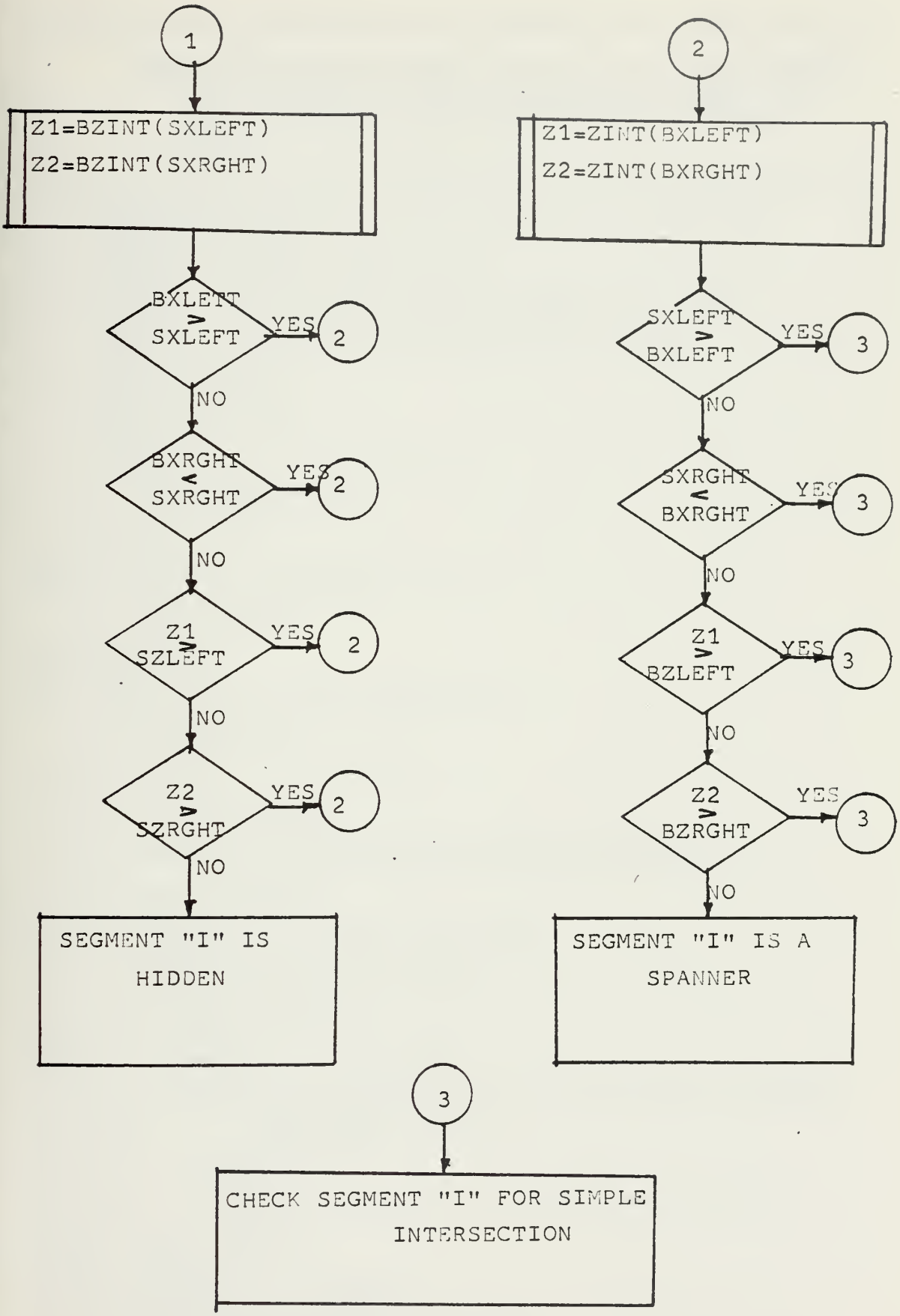
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       SHOWIN: DIVIDES THE YS COORDINATE VALUES BY TWO IN ORDER
C           TO DISPLAY AN UN-DISTORTED IMAGE ON THE RAMTEK AT THE
C           NAVAL POST GRADUATE SCHOOL WHICH HAS DOUBLE WIDE
C           HORIZONTAL RESOLUTION LINES.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE SHOWIN(IT)
      COMMON /AAB/ XS(120),YS(120),ZS(120),POINTM
      INTEGER POINTM
      IT=IT/2
      DO 1600 LI=1,POINTM
        YS(LI)=YS(LI)/2.0
1600  CONTINUE
      RETURN
      END
```


SHOWIN(IT)
IT=IT/2
D O
LI=1,POINTM
YS(I)=YS(I)/2
RETURN

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       LOOKER: EXAMINES THE ACTIVE LIST OF SEGMENTS AND DETERMINES
C           HOW MANY ARE VIEWABLE IN THIS SPAN.  IT ALSO GENERATES
C           A BOX AROUND ALL VIEWABLE SEGMENTS AND PASSES THE
C           NUMBER OF SEGMENTS ARE IN THE BOX AND BOX TYPE TO THE
C           THINKR.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE LOOKER(I)
      COMMON /TD/XLEFT(60),XRIGHT(60),ZLEFT(60),ZRIGHT(60)
      COMMON /TE/XSPNLF,XSPNRG,XRESL
      COMMON /TF/IBXCNT,IBXTYP
      COMMON /TG/BXLEFT,BXRGHT,BZLEFT,BZRGHT,BZMIN,BZMAX
      COMMON /TH/SXLEFT,SXRGHT,SZLEFT,SZRGHT
      COMMON /TI/IBSEG1,IBSEG2,DIV,SDIV,IBFULL,ISFULL
      SXLEFT=XLEFT(I)
      SXRGHT=XRIGHT(I)
      SZLEFT=ZLEFT(I)
      SZRGHT=ZRIGHT(I)
      ISFULL=1
      IF(SXLEFT.GT.XSPNLF) GO TO 611
        SZLEFT=ZINT(XSPNLF)
        SXLEFT=XSPNLF
        GO TO 612
611   ISFULL=0
612   IF(SXRGHT.LT.XSPNRG) GO TO 613
        SZRGHT=ZINT(XSPNRG)
        SXRGHT=XSPNRG
        GO TO 614
613   ISFULL=0
614   SDIV=SXLEFT
      IF(SXLEFT.LE.XSPNLF) SDIV=SXRGHT
      IF(IBXCNT.NE.0) GO TO 615
        CALL LOADBX(I)
        RETURN
615   IF(IBXCNT.NE.1) GO TO 616
        Z1=BZINT(SXLEFT)
        Z2=BZINT(SXRGHT)
        IF((BXLEFT.LE.SXLEFT).AND.(BXRGHT.GE.SXRGHT).AND.
     &      (Z1.LE.SZLEFT).AND.(Z2.LE.SZRGHT)) RETURN
        Z1=ZINT(BXLEFT)
        Z2=ZINT(BXRGHT)
        IF(SXLEFT.GT.BXLEFT) GO TO 618
        IF(SXRGHT.LT.BXRGHT) GO TO 618
        IF(Z1.GT.BZLEFT) GO TO 618
        IF(Z2.GT.BZRGHT) GO TO 618
            CALL LOADBX(I)
            RETURN
618     IF(ISFULL.EQ.0) GO TO 619
        IF(IBFULL.EQ.0) GO TO 619
            TEMP=BXLEFT+(BXRGHT-BXLEFT)*(SZLEFT-BZLEFT)/(BZRGHT-
     &          BZLEFT-SZRGHT+SZLEFT)
            CALL XPNDBX(I)
            IBXTYP=1
            DIV=TEMP
            IF(BZLEFT.LT.SZLEFT) CALL ISWAP(IBSEG1,IBSEG2)
            RETURN
619     CALL XPNDBX(I)
        RETURN
616   IF(IBXCNT.LE.1) RETURN
        IF(SXLEFT.GT.BXLEFT) GO TO 620
        IF(SXRGHT.LT.BXRGHT) GO TO 620
```

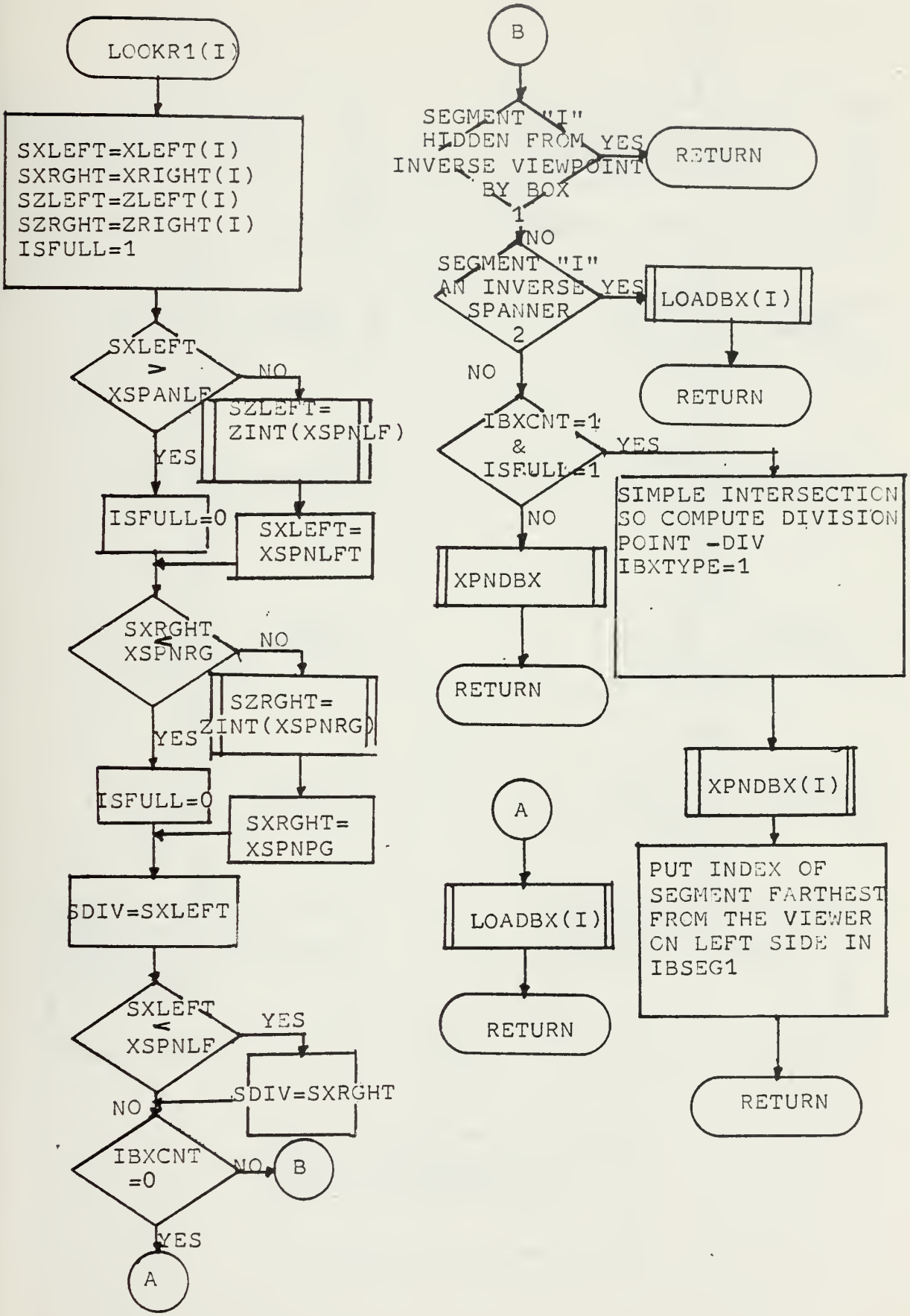
```
      IF(SZLEFT.GT.BZMIN) GO TO 620
      IF(SZRGHT.GT.BZMIN) GO TO 620
          CALL LOADBX(I)
          RETURN
620   IF(BXLEFT.GT.SXLEFT) GO TO 621
      IF(BXRGHT.LT.SXRGHT) GO TO 621
      IF(BZMAX.GT.SZLEFT) GO TO 621
      IF(BZMAX.GT.SZRGHT) GO TO 621
      RETURN
621   CALL XPNDBX(I)
      RETURN
      END
```

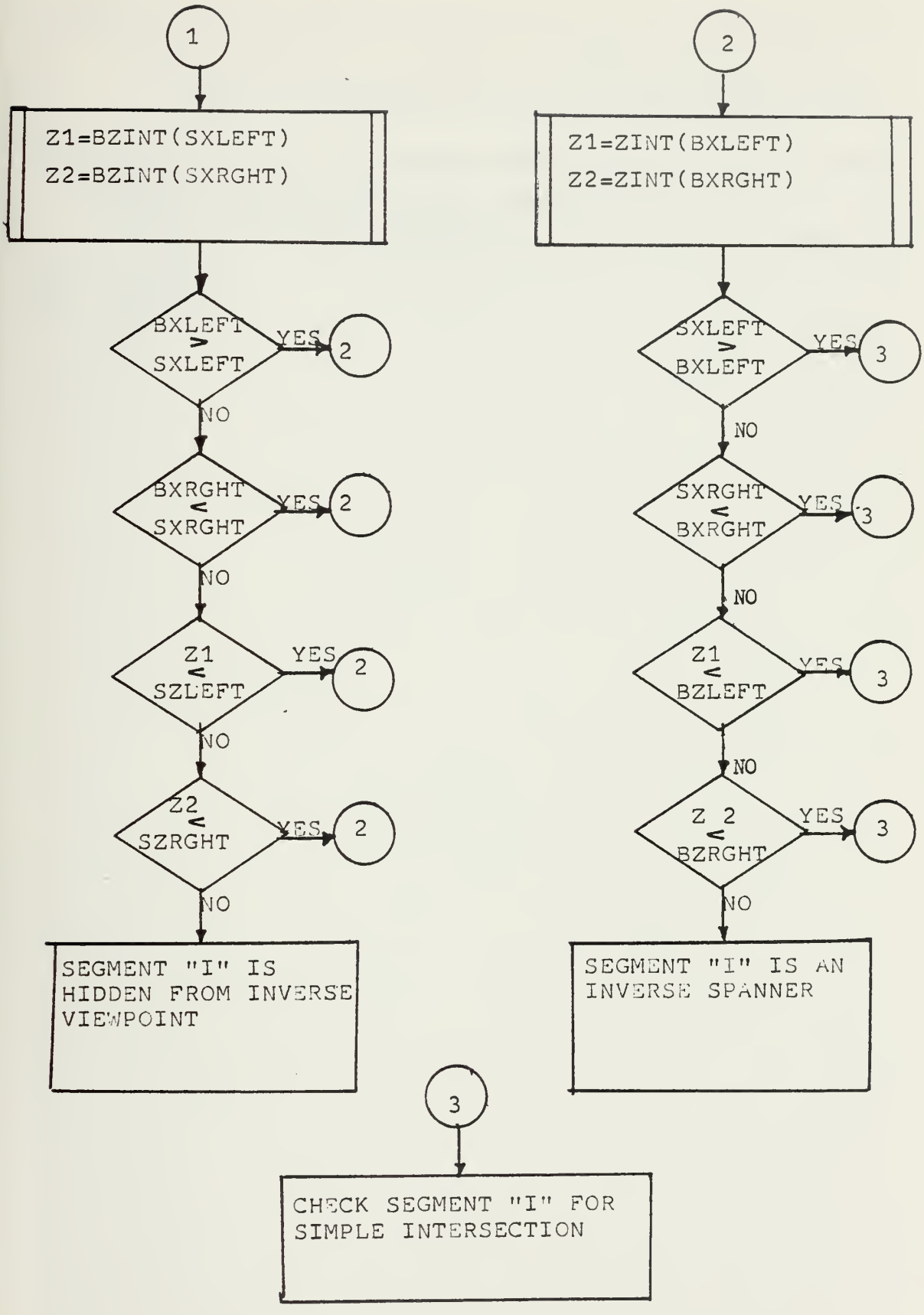

LOOKER(I)
SXLEFT=XLEFT(I)
SXRGHT=XRIGHT(I)
SZLEFT=ZLEFT(I)
SZRGHT=ZRIGHT(I)
ISFULL=1
SXLEFT < XSPANLF
NO
YES
SZLEFT= ZINT(XSPNLF)
ISFULL=0
SXLEFT= XSPNLFT
SXRGHT < XSPNRG
NO
YES
SZRGHT= ZINT(XSPNRG)
ISFULL=0
SXRGHT= XSPNPG
SDIV=SXLEFT
SXLEFT ≥ XSPNLF
YES
NO
SDIV=SXRGHT
IBXCNT =0
NO
B
YES
A
B
SEGMENT "I" HIDDEN FROM VIEWPOINT BY BOX 1
YES
RETURN
NO
SEGMENT "I" A SPANNER 2
YES
LOADBX(I)
RETURN
NO
IBXCNT=1 & ISFULL
YES
NO
XPNDBX(I)
RETURN
SIMPLE INTERSECTION SO COMPUTE DIVISION POINT-DIYIBXTYP=1
XPNDBX(I)
PUT INDEX OF SEGMENT CLOSEST TO VIEWER ON LEFT SIDE IN IBSEG1
RETURN
A
LOADBX(I)
RETURN


1
Z1=BZINT(SXLEFT)
Z2=BZINT(SXRGHT)
BXLETT > SXLEFT
YES
2
NO
BXRGHT < SXRGHT
YES
2
NO
Z1 > SZLEFT
YES
2
NO
Z2 > SZRGHT
YES
2
NO
SEGMENT "I" IS HIDDEN
2
Z1=ZINT(BXLEFT)
Z2=ZINT(BXRGHT)
SXLEFT > BXLEFT
YES
3
NO
SXRGHT < BXRGHT
YES
3
NO
Z1 > BZLEFT
YES
3
NO
Z2 > BZRGHT
YES
3
NO
SEGMENT "I" IS A SPANNER
3
CHECK SEGMENT "I" FOR SIMPLE INTERSECTION

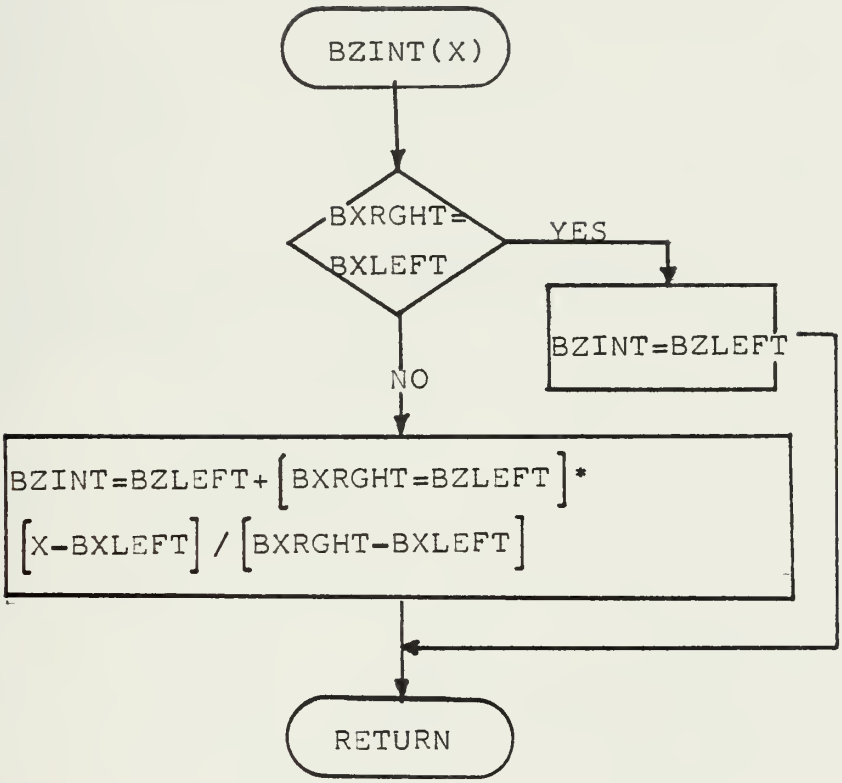
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       LOOKR1: EXAMINES THE ACTIVE LIST OF SEGMENTS AND DETERMINES
C           HOW MANY ARE VIEWABLE IN THIS SPAN.  IT ALSO GENERATES
C           A BOX AROUND ALL VIEWABLE SEGMENTS AND PASSES THE
C           NUMBER OF SEGMENTS ARE IN THE BOX AND BOX TYPE TO THE
C           THINKR.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE LOOKR1(I)
      COMMON /TD/XLEFT(60),XRIGHT(60),ZLEFT(60),ZRIGHT(60)
      COMMON /TE/XSPNLF,XSPNRG,XRESL
      COMMON /TF/IBXCNT,IBXTYP
      COMMON /TG/BXLEFT,BXRGHT,BZLEFT,BZRGHT,BZMIN,BZMAX
      COMMON /TH/SXLEFT,SXRGHT,SZLEFT,SZRGHT
      COMMON /TI/IBSEG1,IBSEG2,DIV,SDIV,IBFULL,ISFULL
      SXLEFT=XLEFT(I)
      SXRGHT=XRIGHT(I)
      SZLEFT=ZLEFT(I)
      SZRGHT=ZRIGHT(I)
      ISFULL=1
      IF(SXLEFT.GT.XSPNLF) GO TO 611
        SZLEFT=ZINT(XSPNLF)
        SXLEFT=XSPNLF
        GO TO 612
611   ISFULL=0
612   IF(SXRGHT.LT.XSPNRG) GO TO 613
        SZRGHT=ZINT(XSPNRG)
        SXRGHT=XSPNRG
        GO TO 614
613   ISFULL=0
614   SDIV=SXLEFT
      IF(SXLEFT.LE.XSPNLF) SDIV=SXRGHT
      IF(IBXCNT.NE.0) GO TO 615
        CALL LOADBX(I)
        RETURN
615   IF(IBXCNT.NE.1) GO TO 616
        Z1=BZINT(SXLEFT)
        Z2=BZINT(SXRGHT)
        IF((BXLEFT.LE.SXLEFT).AND.(BXRGHT.GE.SXRGHT).AND.
     &      (Z1.GE.SZLEFT).AND.(Z2.GE.SZRGHT)) RETURN
        Z1=ZINT(BXLEFT)
        Z2=ZINT(BXRGHT)
        IF(SXLEFT.GT.BXLEFT) GO TO 618
        IF(SXRGHT.LT.BXRGHT) GO TO 618
        IF(Z1.GT.BZLEFT) GO TO 618
        IF(Z2.GT.BZRGHT) GO TO 618
            CALL LOADBX(I)
            RETURN
618     IF(ISFULL.EQ.0) GO TO 619
        IF(IBFULL.EQ.0) GO TO 619
            TEMP=BXLEFT+(BXRGHT-BXLEFT)*(SZLEFT-BZLEFT)/(BZRGHT-
     &          BZLEFT-SZRGHT+SZLEFT)
            CALL XPNDBX(I)
            IBXTYP=1
            DIV=TEMP
            IF(BZLEFT.GT.SZLEFT) CALL ISWAP(IBSEG1,IBSEG2)
            RETURN
619     CALL XPNDBX(I)
        RETURN
616   IF(IBXCNT.LE.1) RETURN
        IF(SXLEFT.GT.BXLEFT) GO TO 620
        IF(SXRGHT.LT.BXRGHT) GO TO 620
```

```
      IF(SZLEFT.LT.BZMAX) GO TO 620
      IF(SZRGHT.LT.BZMAX) GO TO 620
          CALL LOADBX(I)
          RETURN
620   IF(BXLEFT.GT.SXLEFT) GO TO 621
      IF(BXRGHT.LT.SXRGHT) GO TO 621
      IF(BZMIN.LT.SZLEFT) GO TO 621
      IF(BZMIN.LT.SZRGHT) GO TO 621
      RETURN
621   CALL XPNDBX(I)
      RETURN
      END
```

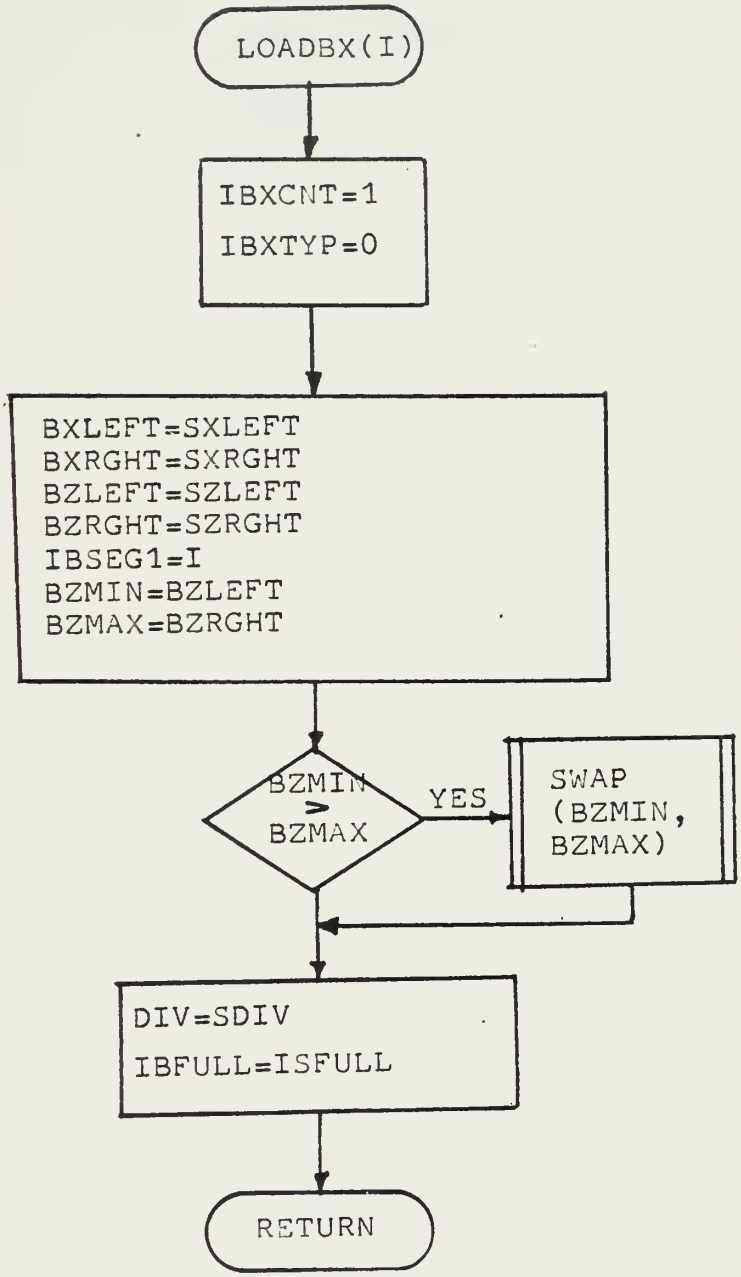

LOOKR1(I)
SXLEFT=XLEFT(I)
SXRGHT=XRIGHT(I)
SZLEFT=ZLEFT(I)
SZRGHT=ZRIGHT(I)
ISFULL=1
SXLEFT > XSPANLF
NO
YES
SZLEFT= ZINT(XSPNLF)
ISFULL=0
SXLEFT= XSPNLFT
SXRGHT < XSPNRG
NO
YES
SZRGHT= ZINT(XSPNRG)
ISFULL=0
SXRGHT= XSPNPG
SDIV=SXLEFT
SXLEFT < XSPNLF
YES
NO
SDIV=SXRGHT
IBXCNT =0
NO
B
YES
A
B
SEGMENT "I" HIDDEN FROM INVERSE VIEWPOINT BY BOX 1
YES
RETURN
NO
SEGMENT "I" AN INVERSE SPANNER 2
YES
LOADBX(I)
RETURN
NO
IBXCNT=1 & ISFULL=1
YES
NO
XPNDBX
RETURN
SIMPLE INTERSECTION SO COMPUTE DIVISION POINT -DIV
IBXTYPE=1
XPNDBX(I)
PUT INDEX OF SEGMENT FARTHEST FROM THE VIEWER ON LEFT SIDE IN IBSEG1
RETURN
A
LOADBX(I)
RETURN


1
Z1=BZINT(SXLEFT)
Z2=BZINT(SXRGHT)
BXLEFT > SXLEFT
YES
2
NO
BXRGHT < SXRGHT
YES
2
NO
Z1 < SZLEFT
YES
2
NO
Z2 < SZRGHT
YES
2
NO
SEGMENT "I" IS HIDDEN FROM INVERSE VIEWPOINT
2
Z1=ZINT(BXLEFT)
Z2=ZINT(BXRGHT)
SXLEFT > BXLEFT
YES
3
NO
SXRGHT < BXRGHT
YES
3
NO
Z1 < BZLEFT
YES
3
NO
Z 2 < BZRGHT
YES
3
NO
SEGMENT "I" IS AN INVERSE SPANNER
3
CHECK SEGMENT "I" FOR SIMPLE INTERSECTION

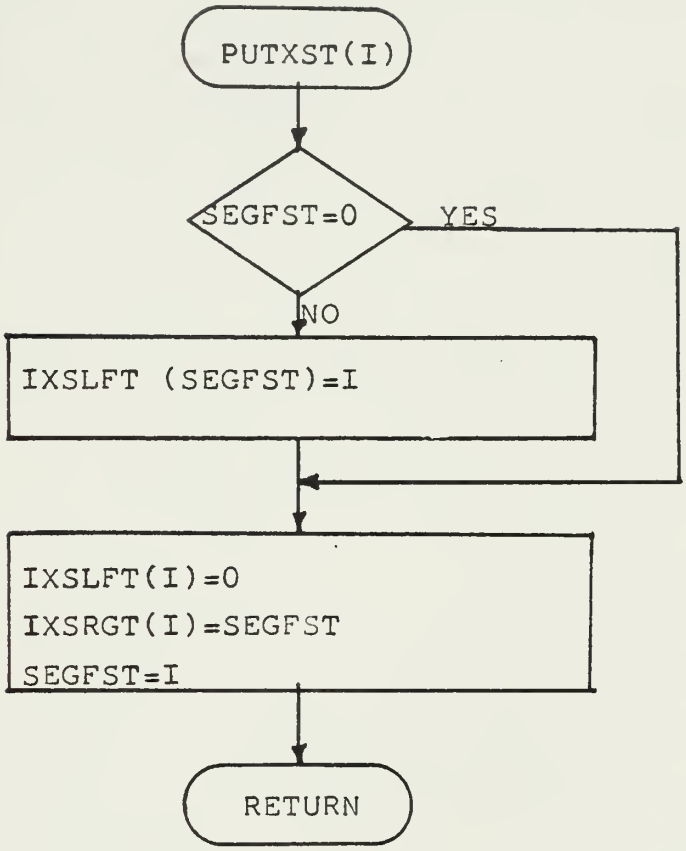
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       BZINT: IS USED BY THE LOOKER TO DETERMINE THE DEPTH (THE ZS
C           VALUE) OF THE SEGMENT BOX AT ANY XS VALUE WITHIN THIS
C           SPAN.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      FUNCTION BZINT(X)
      COMMON /TG/BXLEFT,BXRGHT,BZLEFT,BZRGHT,BZMIN,BZMAX
      IF(BXRGHT.EQ.BXLEFT) GO TO 621
        BZINT=BZLEFT+(BZRGHT-BZLEFT)*(X-BXLEFT)/(BXRGHT-BXLEFT)
        RETURN
621   BZINT=BZLEFT
      RETURN
      END
```


BZINT(X)
BXRGHT=
BXLEFT
YES
BZINT=BZLEFT
NO
BZINT=BZLEFT+⌊BXRGHT=BZLEFT⌋*
⌊X-BXLEFT⌋/⌊BXRGHT-BXLEFT⌋
RETURN

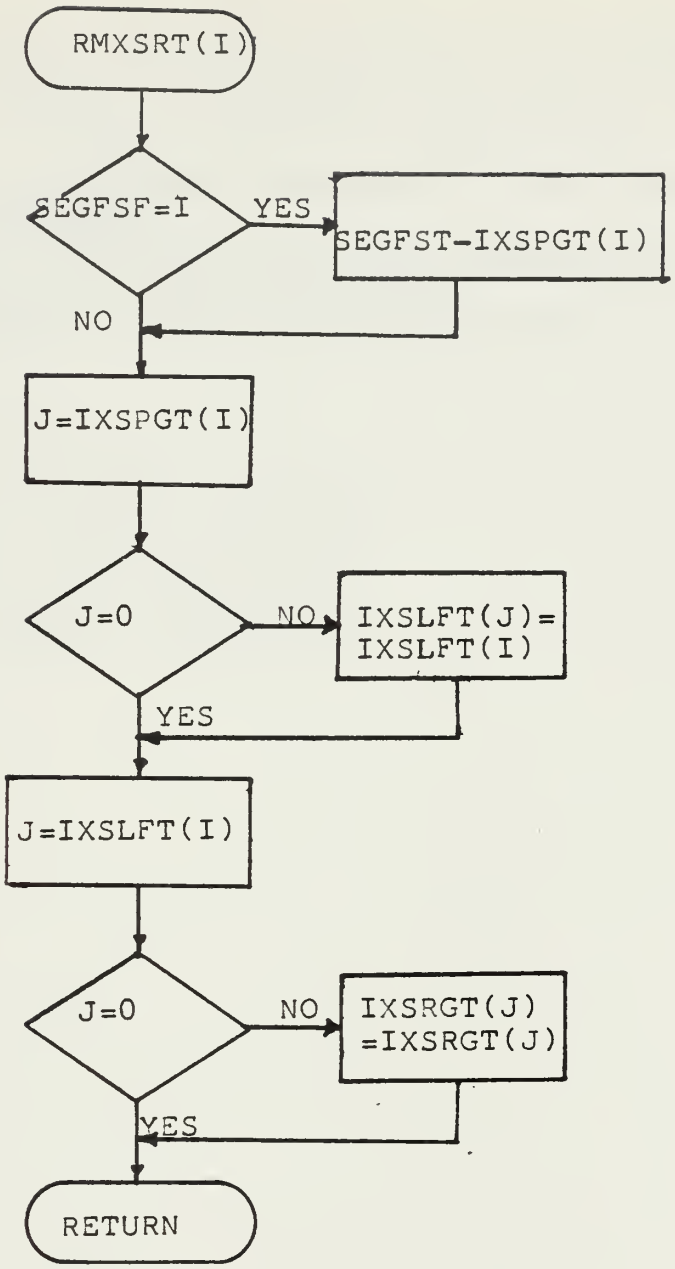
```
CCCC
```


LOADBX(I)
IBXCNT=1
IBXTYP=0
BXLEFT=SXLEFT
BXRGHT=SXRGHT
BZLEFT=SZLEFT
BZRGHT=SZRGHT
IBSEG1=I
BZMIN=BZLEFT
BZMAX=BZRGHT
BZMIN > BZMAX
YES
SWAP (BZMIN, BZMAX)
DIV=SDIV
IBFULL=ISFULL
RETURN

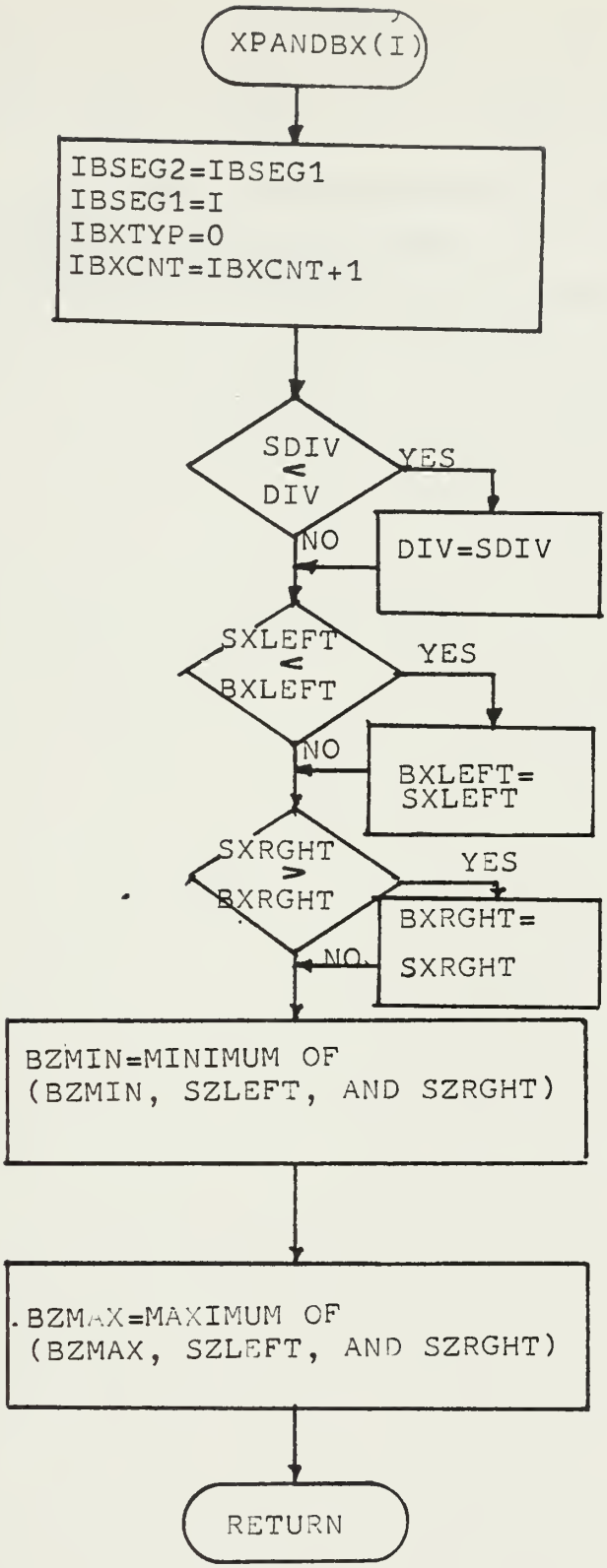
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       PUTXST: PLACES THE SEGMENT PASSED TO IT AS THE CALLING
C           PARAMETER I INTO OF THE XSORT LISTS, IXSLFT(I)
C           AND IXSRGT(I).
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE PUTXST(I)
      COMMON /TC/IXSLFT(60),IXSRGT(60),SEGFST
      INTEGER SEGFST
      IF(SEGFST.NE.0) IXSLFT(SEGFST)=I
      IXSLFT(I)=0
      IXSRGT(I)=SEGFST
      SEGFST=I
      RETURN
      END
```


PUTXST(I)
SEGFST=0
YES
NO
IXSLFT (SEGFST)=I
IXSLFT(I)=0
IXSRGT(I)=SEGFST
SEGFST=I
RETURN

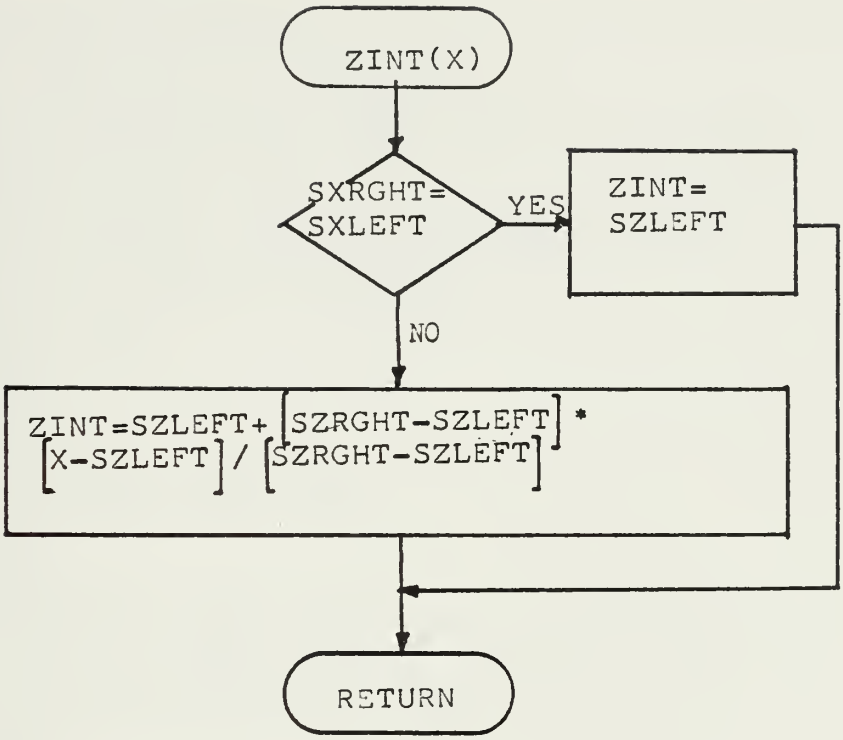
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       RMXSRT: REMOVES A SEGMENT FROM THE XSORT LISTS, IXSRGT(I)
C           AND IXSLFT(I).
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE RMXSRT(I)
      COMMON /TC/IXSLFT(60),IXSRGT(60),SEGFST
      INTEGER SEGFST
      IF(SEGFST.EQ.I)SEGFST=IXSRGT(I)
      J=IXSRGT(I)
      IF(J.NE.0) IXSLFT(J)=IXSLFT(I)
      J=IXSLFT(I)
      IF(J.NE.0) IXSRGT(J)=IXSRGT(I)
      RETURN
      END
```


RMXSRT(I)
SEGFSF=I
YES
SEGFST-IXSPGT(I)
NO
J=IXSPGT(I)
J=0
NO
IXSLFT(J)=
IXSLFT(I)
YES
J=IXSLFT(I)
J=0
NO
IXSRGT(J)
=IXSRGT(J)
YES
RETURN

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       XPNDBX: IS CALLED BY THE LOOKER TO XPAND THE BOX SURROUNDING
C           THE VIEWABLE SEGMENTS IN THE CURRENT SPAN.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE XPNDBX(I)
      COMMON /TF/IBXCNT,IBXTYP
      COMMON /TG/BXLEFT,BXRGHT,BZLEFT,BZRGHT,BZMIN,BZMAX
      COMMON /TH/SXLEFT,SXRGHT,SZLEFT,SZRGHT
      COMMON /TI/IBSEG1,IBSEG2,DIV,SDIV,IBFULL,ISFULL
      IBSEG2=IBSEG1
      IBSEG1=I
      IBXTYP=0
      IBXCNT=IBXCNT+1
      IF(SDIV.LT.DIV)DIV=SDIV
      IF(SXLEFT.LT.BXLEFT) BXLEFT=SXLEFT
      IF(SXRGHT.GT.BXRGHT) BXRGHT=SXRGHT
      IF(SZLEFT.LT.BZMIN) BZMIN=SZLEFT
      IF(SZRGHT.LT.BZMIN) BZMIN=SZRGHT
      IF(SZLEFT.GT.BZMAX) BZMAX=SZLEFT
      IF(SZRGHT.GT.BZMAX) BZMAX=SZRGHT
      RETURN
      END
```

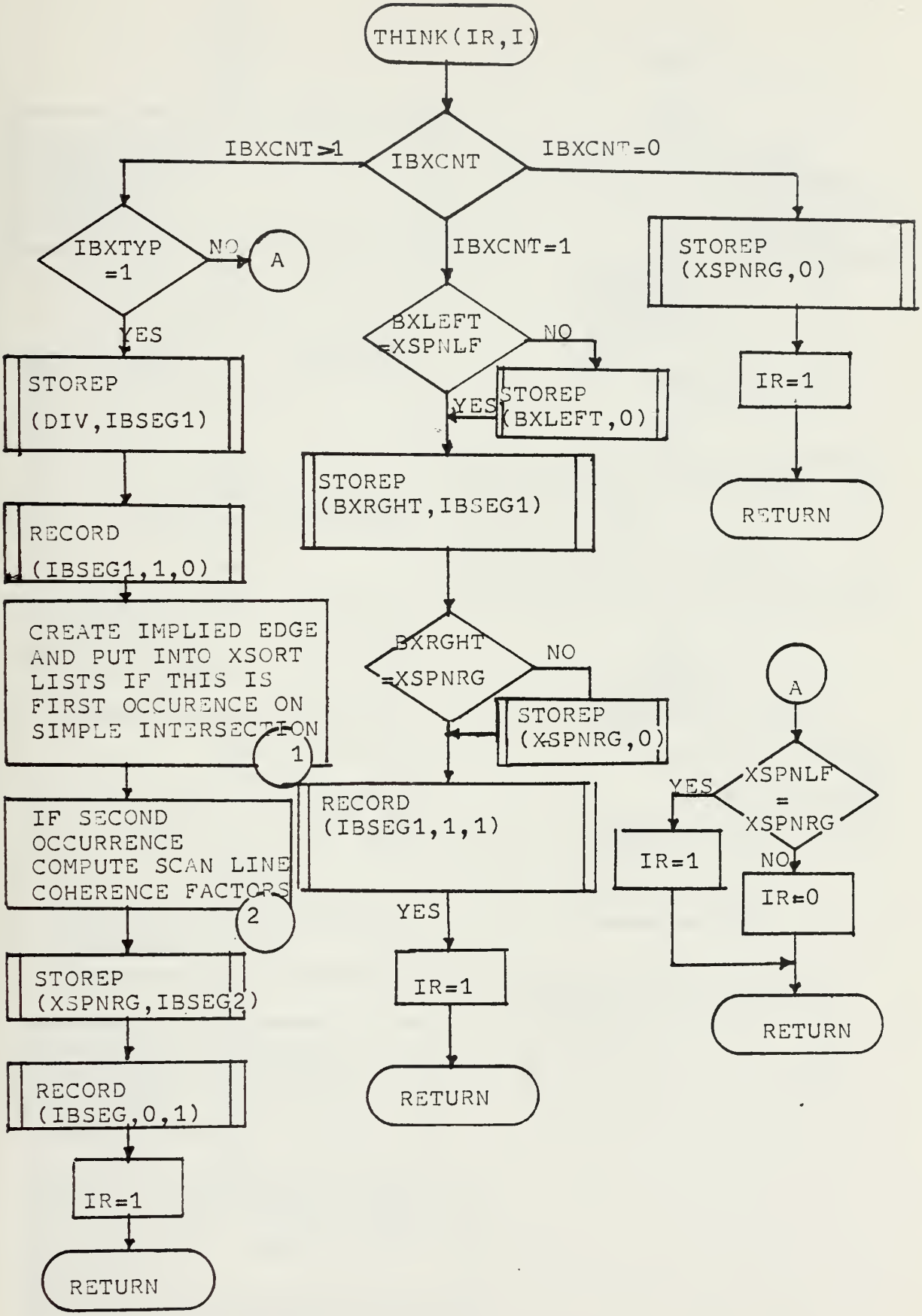

XPANDBX(I)
IBSEG2=IBSEG1
IBSEG1=I
IBXTYP=0
IBXCNT=IBXCNT+1
SDIV < DIV
YES
NO
DIV=SDIV
SXLEFT < BXLEFT
YES
NO
BXLEFT= SXLEFT
SXRGHT > BXRGHT
YES
NO
BXRGHT= SXRGHT
BZMIN=MINIMUM OF (BZMIN, SZLEFT, AND SZRGHT)
BZMAX=MAXIMUM OF (BZMAX, SZLEFT, AND SZRGHT)
RETURN

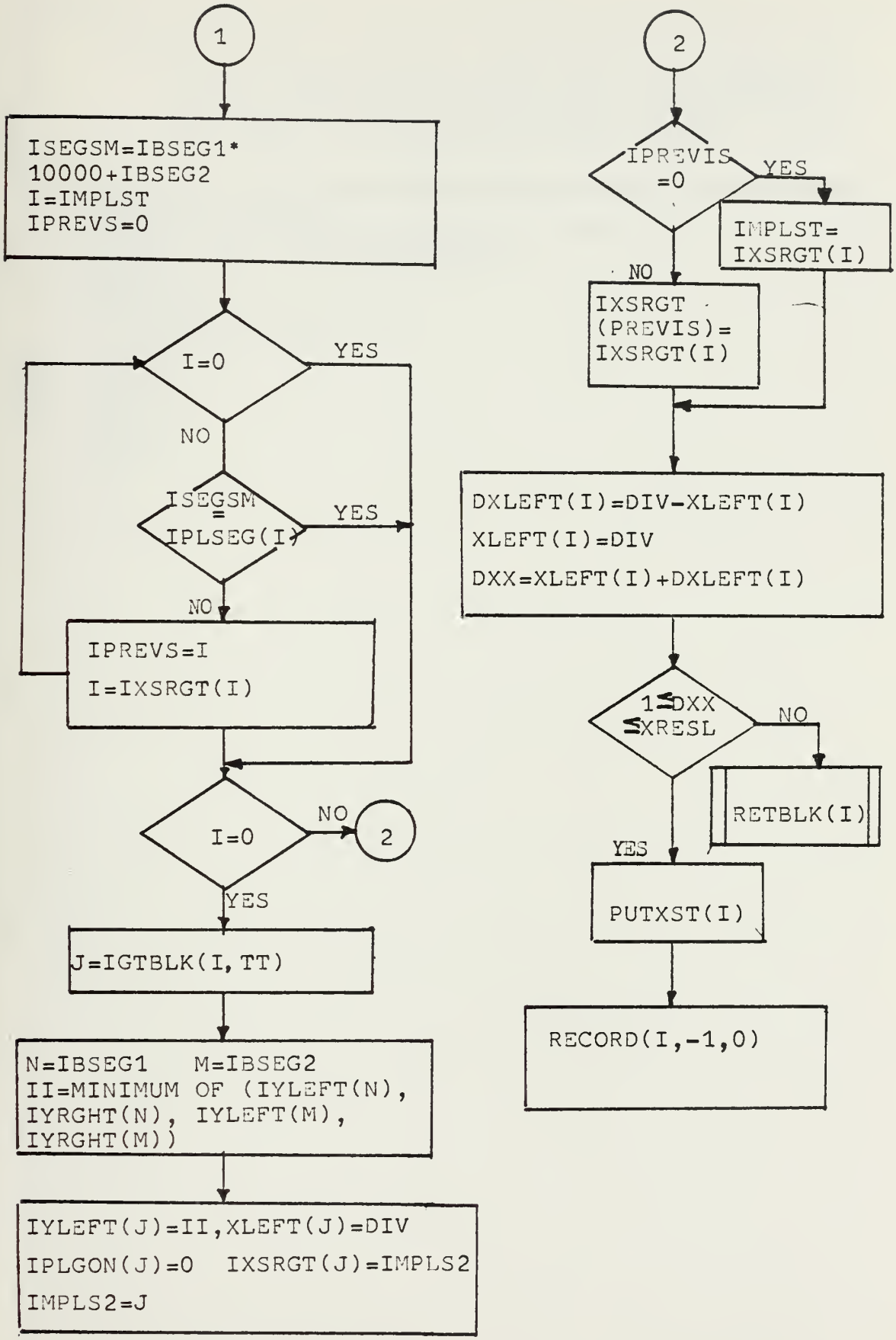
```
CCCC
```


ZINT(X)
SXRGHT=
SXLEFT
YES
ZINT=
SZLEFT
NO
ZINT=SZLEFT+ ⌈SZRGHT-SZLEFT⌉ *
⌈X-SZLEFT⌉ / ⌈SZRGHT-SZLEFT⌉
RETURN

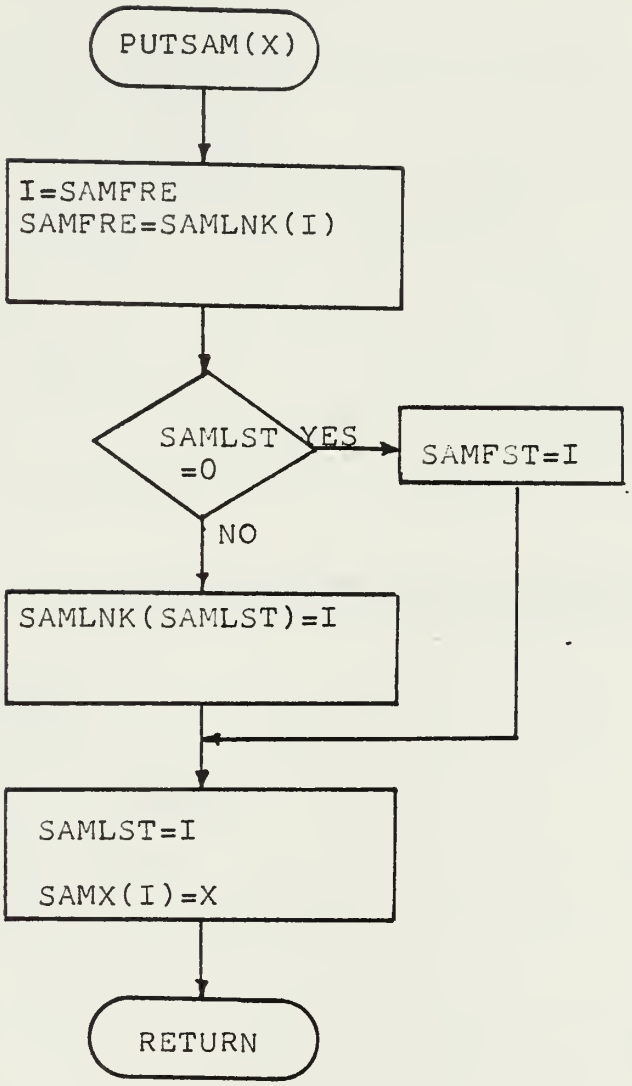
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       THINKR: RECEIVES THE INFORMATION ABOUT THE CONTENTS OF
C           THIS SPAN WHICH WAS DEVELOPED BY THE LOOKER AND
C           DETERMINES IF THIS SPAN CONTAINS DISPLAYABLE
C           INFORMATION.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE THINKR(IR,I)
      COMMON /TB/IYLEFT(60),IYRGHT(60)
      COMMON /TC/IXSLFT(60),IXSRGT(60),SEGFST
      COMMON /TD/XLEFT(60),XRIGHT(60),ZLEFT(60),ZRIGHT(60)
      COMMON /TE/XSPNLF,XSPNRG,XRESL
      COMMON /TF/IBXCNT,IBXTYP
      COMMON /TG/BXLEFT,BXRGHT,BZLEFT,BZRGHT,BZMIN,BZMAX
      COMMON /TI/IBSEG1,IBSEG2,DIV,SDIV,IFULL,ISFULL
      COMMON /TJ/ISEGSM,IMPLST,IMPLS2,IPREVS
      COMMON /TK/IPLSEG(60),IPLGON(60)
      COMMON /TL/DXLEFT(60),DXRGHT(60)
      INTEGER SEGFST
      IZERO=0
      IMIN=-1
      IONE=1
      IONE1=1
      IF(IBXCNT.NE.0) GO TO 623
        CALL STOREP(XSPNRG,IZERO)
        IR=1
        RETURN
623   IF(IBXCNT.NE.1) GO TO 624
        IF(BXLEFT.NE.XSPNLF) CALL STOREP(BXLEFT,IZERO)
        CALL STOREP(BXRGHT,IBSEG1)
        IF(BXRGHT.NE.XSPNRG) CALL STOREP(XSPNRG,IZERO)
        CALL RECORD(IBSEG1,IONE,IONE1)
        IR=1
        RETURN
624   IF(IBXTYP.NE.1) GO TO 625
        CALL STOREP(DIV,IBSEG1)
        CALL RECORD(IBSEG1,IONE,IZERO)
        ISEGSM=IBSEG1*10000+IBSEG2
        I=IMPLST
        IPREVS=0
627     IF(I.EQ.0) GO TO 626
        IF(ISEGSM.EQ.IPLSEG(I)) GO TO 626
            IPREVS=I
            I=IXSRGT(I)
            GO TO 627
626     IF(I.EQ.0) GO TO 628
        IF(IPREVS.EQ.0) GO TO 629
            IXSRGT(IPREVS)=IXSRGT(I)
            GO TO 630
629     IMPLST=IXSRGT(I)
630     DXLEFT(I)=DIV-XLEFT(I)
        XLEFT(I)=DIV
        DXX=XLEFT(I)+DXLEFT(I)
        IF(DXX.LT.1) GO TO 631
        IF(DXX.GT.XRESL) GO TO 631
            CALL PUTXST(I)
            CALL RECORD(I,IMIN,IZERO)
            GO TO 632
631     CALL RFIBLK(I)
        GO TO 632
628     J=IGTBLK(ITT)
        IF(J.EQ.0) GO TO 160
```

```
            IPLSEG(J)=ISEGSM
            II=IYLEFT(IBSEG1)
            IF(II.LT.IYRGHT(IBSEG1)) II=IYRGHT(IBSEG1)
            IF(II.LT.IYLEFT(IBSEG2)) II=IYLEFT(IBSEG2)
            IF(II.LT.IYRGHT(IBSEG2)) II=IYRGHT(IBSEG2)
            IYLEFT(J)=II
            IPLGON(J)=0
            XLEFT(J)=DIV
            IXSRGT(J)=IMPLS2
            IMPLS2=J
632       CALL STOREP(XSPNRG,IBSEG2)
          CALL RECORD(IBSEG2,IZERO,IONE)
          IR=1
          RETURN
625     IF(XSPNLF.EQ.XSPNRG) GO TO 633
          IR=0
          RETURN
633     IR=1
        RETURN
  160   WRITE(6,999)
  999   FORMAT(2X,'THE NUMBER OF SEGMENTS HAS EXCEEDED THE STORAGE
       &PROVIDED')
        IR=2
        END
```

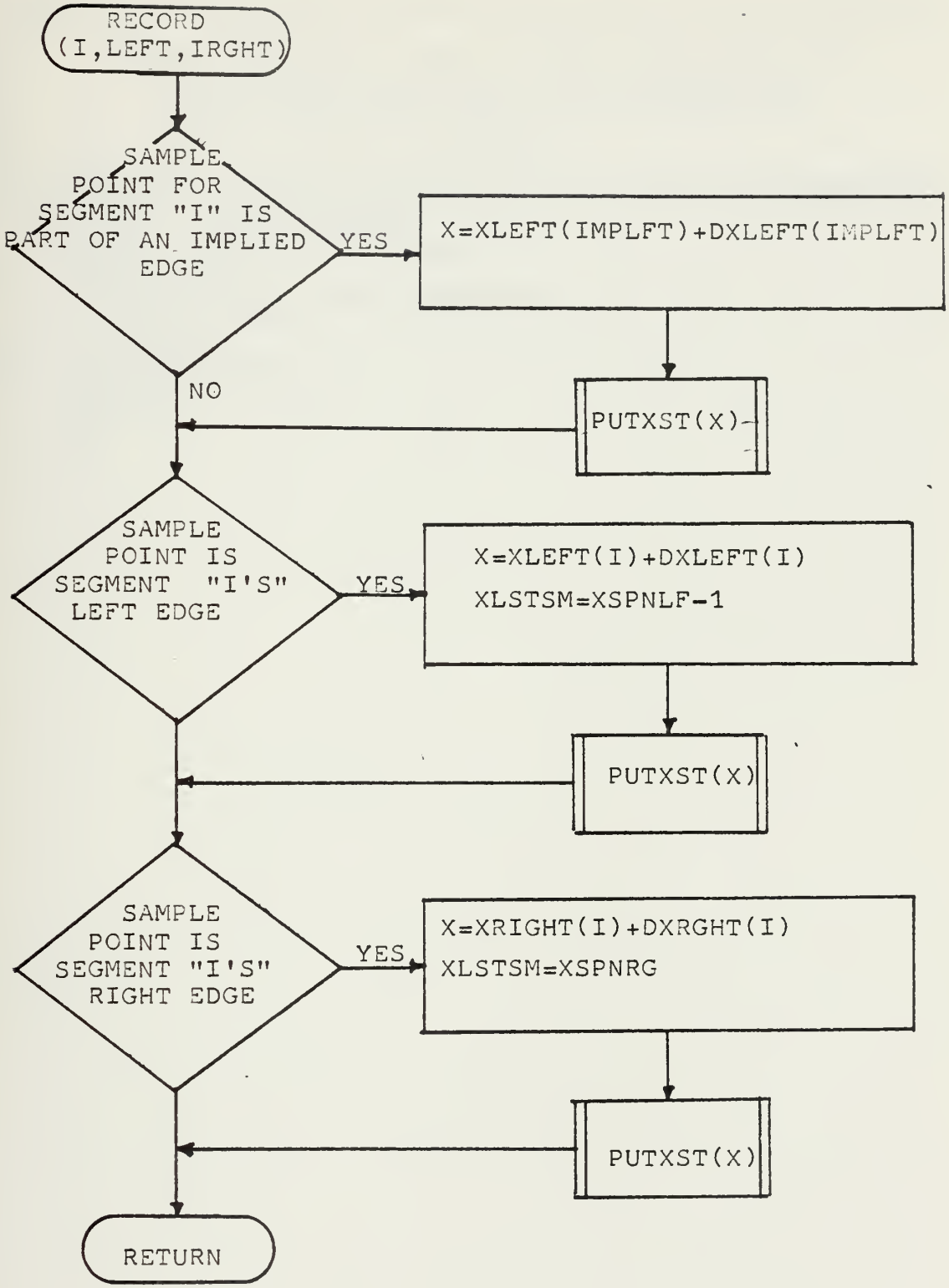

THINK(IR,I)
IBXCNT
IBXCNT>1
IBXCNT=0
IBXCNT=1
IBXTYP =1
NO
A
YES
STOREP (DIV,IBSEG1)
RECORD (IBSEG1,1,0)
CREATE IMPLIED EDGE AND PUT INTO XSORT LISTS IF THIS IS FIRST OCCURENCE ON SIMPLE INTERSECTION
1
IF SECOND OCCURRENCE COMPUTE SCAN LINE COHERENCE FACTORS
2
STOREP (XSPNRG,IBSEG2)
RECORD (IBSEG,0,1)
IR=1
RETURN
BXLEFT =XSPNLF
NO
YES
STOREP (BXLEFT,0)
STOREP (BXRGHT,IBSEG1)
BXRGHT =XSPNRG
NO
STOREP (XSPNRG,0)
RECORD (IBSEG1,1,1)
YES
IR=1
RETURN
STOREP (XSPNRG,0)
IR=1
RETURN
A
XSPNLF = XSPNRG
YES
IR=1
NO
IR=0
RETURN


1
ISEGSM=IBSEG1*
10000+IBSEG2
I=IMPLST
IPREVS=0
I=0
YES
NO
ISEGSM
=
IPLSEG(I)
YES
NO
IPREVS=I
I=IXSRGT(I)
I=0
NO
2
YES
J=IGTBLK(I,TT)
N=IBSEG1 M=IBSEG2
II=MINIMUM OF (IYLEFT(N),
IYRGHT(N), IYLEFT(M),
IYRGHT(M))
IYLEFT(J)=II,XLEFT(J)=DIV
IPLGON(J)=0 IXSRGT(J)=IMPLS2
IMPLS2=J
2
IPREVIS
=0
YES
IMPLST=
IXSRGT(I)
NO
IXSRGT
(PREVIS)=
IXSRGT(I)
DXLEFT(I)=DIV-XLEFT(I)
XLEFT(I)=DIV
DXX=XLEFT(I)+DXLEFT(I)
1≤DXX
≤XRESL
NO
RETBLK(I)
YES
PUTXST(I)
RECORD(I,-1,0)

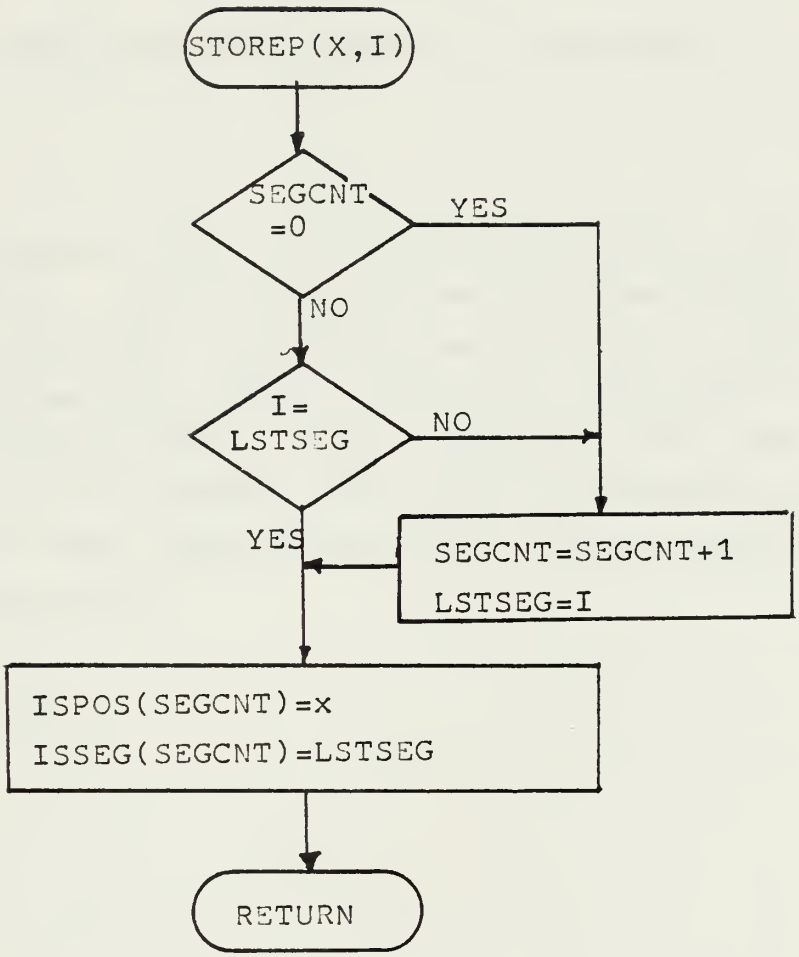
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       PUTSAM: STORES THE SUCESSFUL SAMPLE POINTS (THE SCAN LINE
C           DIVISION POINTS) OF THE PRESENT SCAN LINE WHICH WILL
C           BE USED TO DIVIDE THE NEXT SCAN LINE.  THIS REDUCES THE
C           TIME REQUIRED TO PROCESS AND DISPLAY AN IMAGE.  THIS
C           PROCEEDURE IS CALLED BY SUBROUTINE RECORD.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE PUTSAM(X)
      COMMON /TM/SAMFRE,SAMLST,SAMLNK(120),SAMX(120)
      INTEGER SAMFRE,SAMLST,SAMLNK,SAMX
      I=SAMFRE
      SAMFRE=SAMLNK(I)
      IF(SAMLST.EQ.0) GO TO 634
        SAMLNK(SAMLST)=I
        GO TO 635
634   SAMFST=I
635   SAMLST=I
      SAMX(I)=X
      RETURN
      END
```


PUTSAM(X)
I=SAMFRE
SAMFRE=SAMLNK(I)
SAMLST =0
YES
SAMFST=I
NO
SAMLNK(SAMLST)=I
SAMLST=I
SAMX(I)=X
RETURN

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       RECORD: DECIDES WHICH SCAN LINE SAMPLE POINTS SHOULD BE
C           SAVED TO DIVIDE THE NEXT SCAN LINE INTO SPANS.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE RECORD(I,LEFT,IRGHT)
      COMMON /TN/IMPLFT,XLSTSM
      COMMON /TD/XLEFT(60),XRIGHT(60),ZLEFT(60),ZRIGHT(60)
      COMMON /TK/POLSEG(60),POLGON(60)
      COMMON /TL/DXLEFT(60),DXRGHT(60)
      COMMON /TE/IYLEFT(60),IYRGHT(60)
      COMMON /TM/ SAMFRE,SAMLST,SAMLNK(120),SAMX(120)
      COMMON /TE/ XSPNLF,XSPNRG,XREFSL
      INTEGER POLSEG,SAMFRE,SAMLST,SAMLNK,SAMX,POLGON
      IF(LEFT.EQ.0) GO TO 637
      IF(IMPLFT.EQ.0) GO TO 636
      IF(XLEFT(I).GT.XSPNLF) GO TO 636
      J=POLSEG(IMPLFT)/10000
      II=POLSEG(IMPLFT)-J*10000
      IF(I.NE.II) GO TO 636
        X=XLEFT(IMPLFT)+DXLEFT(IMPLFT)
        CALL PUTSAM(X)
        IMPLFT=0
636   IF(IYLEFT(I).GE.-1)GO TO 637
        DEL=XSPNLF-1.0
        XL=XLEFT(I)
        IF((LEFT.NE.-1).AND.((XL.LE.DEL).OR.(XL.GT.XSPNLF))) GO TO 637
        IF((SAMLST.NE.0).AND.(XLSTSM.EQ.DEL).AND.(LEFT.NE.-1))
     &  GO TO 637
            X=XLEFT(I)+DXLEFT(I)
            CALL PUTSAM(X)
            XLSTSM=XSPNLF-1.0
637   IF(IRGHT.EQ.0) RETURN
      IF(IYRGHT(I).GE.-1) RETURN
      IF(XSPNRG.GT.XRIGHT(I)) RETURN
      UP=XSPNRG+1
      IF(XRIGHT(I).GE.UP) RETURN
        X=XRIGHT(I)+DXRGHT(I)
        CALL PUTSAM(X)
        XLSTSM=XSPNRG
      RETURN
      END
```


RECORD
(I,LEFT,IRGHT)
SAMPLE POINT FOR SEGMENT "I" IS PART OF AN IMPLIED EDGE
YES
X=XLEFT(IMPLFT)+DXLEFT(IMPLFT)
NO
PUTXST(X)
SAMPLE POINT IS SEGMENT "I'S" LEFT EDGE
YES
X=XLEFT(I)+DXLEFT(I)
XLSTSM=XSPNLF-1
PUTXST(X)
SAMPLE POINT IS SEGMENT "I'S" RIGHT EDGE
YES
X=XRIGHT(I)+DXRGHT(I)
XLSTSM=XSPNRG
PUTXST(X)
RETURN

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       STOREP: USED BY THE THINKR TO RECORD THE DISPLAY DATA
C           FOR THE CURRENT SCAN LINE.  THE XS VALUES ARE
C           STORED IN THE ARRAY ISPOS(I) AND THE SEGMENT'S
C           INDEX IS STORED IN ISSEG(I).
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE STOREP(X,I)
      COMMON /TO/SEGCNT,LSTSEG
      COMMON /TP/ISPOS(60),ISSEG(60)
      INTEGER SEGCNT
      IF((SEGCNT.NE.0).AND.(I.EQ.LSTSEG)) GO TO 640
        SEGCNT=SEGCNT+1
        LSTSEG=I
640   ISPOS(SEGCNT)=X
      ISSEG(SEGCNT)=LSTSEG
      RETURN
      END
```


STOREP(X,I)
SEGCNT =0
YES
NO
I= LSTSEG
NO
YES
SEGCNT=SEGCNT+1
LSTSEG=I
ISPOS(SEGCNT)=x
ISSEG(SEGCNT)=LSTSEG
RETURN

APPENDIX B

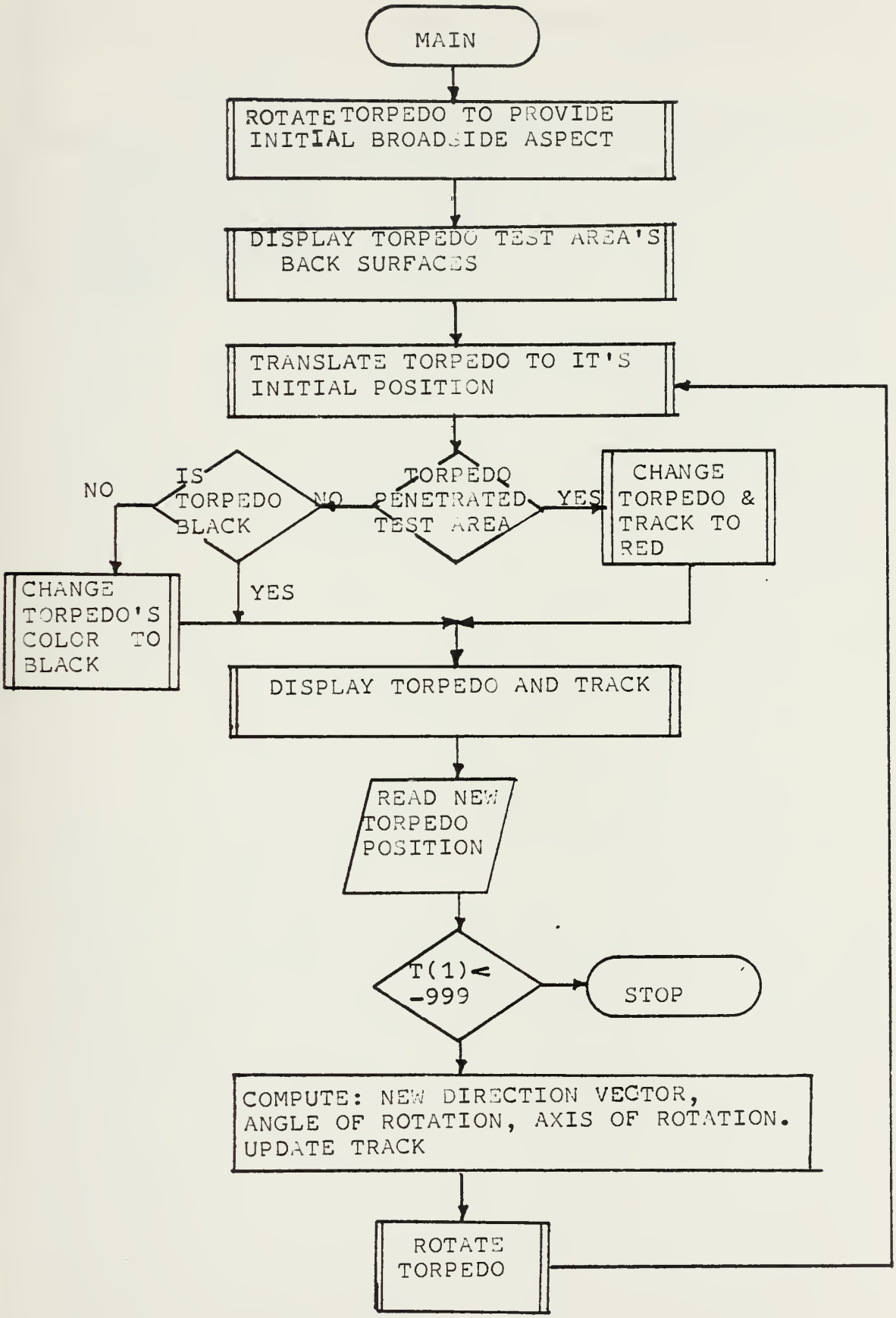
APPLICATIONS PROGRAM AND SUBROUTINES

The applications program and its flow chart were presented first. Next, the seven subroutines which enabled the user to determine polygonal penetration, change the coordinate values of a polygon, and alter the shade or color of a polygon were listed. These programs and their flow charts were not included with the 3-D graphics software package since their applicability was strictly related to a tracking presentation.

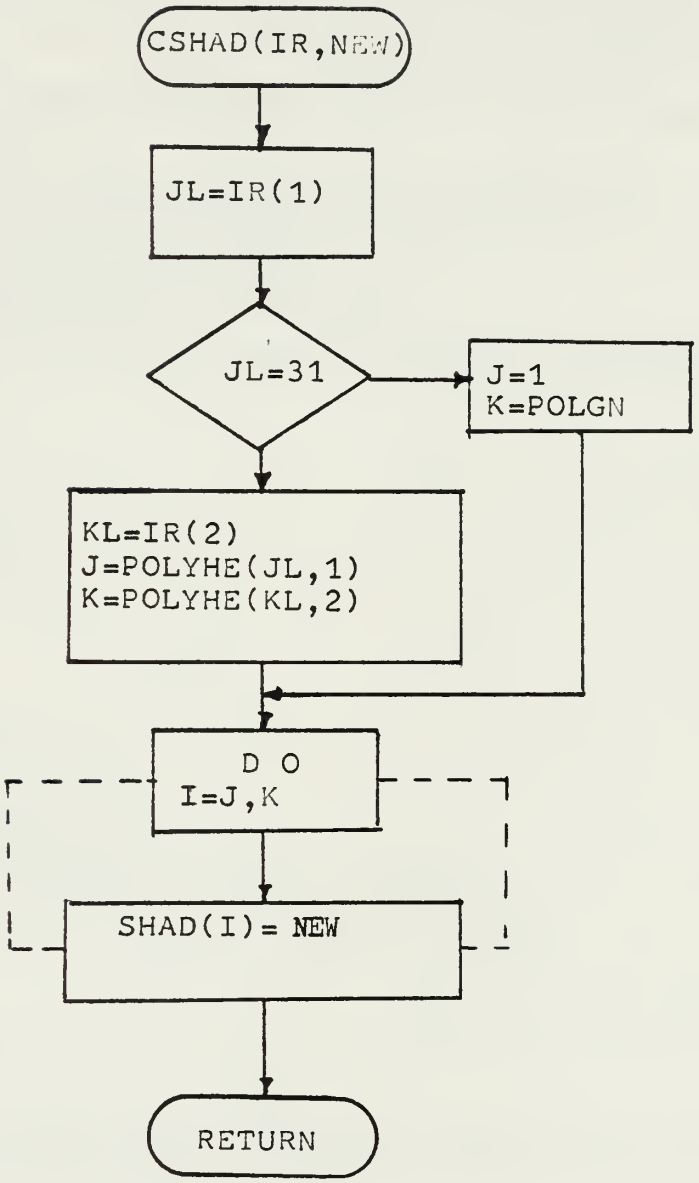
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       THIS IS AN APPLICATION PROGRAM DESIGNED TO DEMONSTRATE THE
C   USAGE OF THE THREE-DIMENSIONAL COMPUTER GRAPHICS SOFTWARE
C   PACKAGE.  THE PROGRAM SIMULATES THE TRACKING OF A TORPEDO IN AN
C   IRREGULAR TORPEDO TEST AREA.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DIMENSION IR(2),S(3),T0(3),T(3),TRACKX(5),TRACKY(5),TRACKZ(5),
     &P2(3),DVECTO(3),N(3),P1(3)
      DATA DVECTO/0.0,-1.0,0.0/
      CALL INITAL
      CONV=180.0/3.14159
      IRED=8
      IPH=5
      ISHAD=16
      NM=0
      I=1
      IR(2)=5
      RISUM=0.0
C
C       INITIALIZING THE OLD DIRECTION VECTOR
C
      DO 80 J=1,3
   80   RISUM=RISUM+DVECTO(J)**2
C
C       MUST SCALE THE IMAGE OF THE TORPEDO SO THAT IT IS VISIBLE
C
      IR(1)=4
      IR(2)=4
      P2(1)=10.0
      P2(2)=10.0
      P2(3)=10.0
      CALL SCALE(IR,P2)
C
C       ROTATE THE VEHICLE SO THAT IT IS INITALLY BROADSIDE TO THE
C         VIEWER.
C
      THETA=90.0
      IAXIS=2
      CALL ROTATE(IR,IAXIS,P1,P2,THETA)
      READ(5,1) T
      DO 10 J=1,3
   10   S(J)=-T(J)
      IAXIS=3
C
C       BEGIN THE RECURSIVE PORTION OF THE PROGRAM
C
  200 TRACKX(I)=T(1)
      TRACKY(I)=T(2)
      TRACKZ(I)=T(3)
      DO 20 J=1,3
   20   T0(J)=T(J)
C
C       DISPLAY THE TORPEDO TEST AREA WITH HIDDEN SURFACES SHOWN.
C
      ILOOK=2
      IR(1)=1
      IR(2)=3
      CALL SURFAC(IR,ILOOK)
C
C       TRANSLATE THE TORPEDO TO ITS PRESENT LOCATION.
C
```

```
      ILOOK=1
      IR(1)=4
      CALL TRANSL(IR,S)
C
C       CHECK THE CURRENT LOCATION OF THE TORPEDO TO
C           DETERMINE IF IT IS WITHIN THE TEST AREA.
C
      IPP=IPENET(IR,T)
      IF(IPP.EQ.1) GO TO 220
      IF(IPP.EQ.0.AND.IFLAG.EQ.1) GO TO 240
  230 IF(NM.LT.5) NM=NM+1
C
C       IF THIS AT LEAST THE SECOND LOCATION OF THE TORPEDO
C           THEN DISPLAY THE TRACK.
C
      IF(NM.GT.1)GO TO 250
C
C       DISPLAY THE TORPEDO WITH THE HIDDEN SURFACES REMOVED.
C
      IR(1)=4
      IR(2)=4
      CALL SURFAC(IR,ILOOK)
C
C       GET THE TORPEDO'S NEW LOCATION.
C
      READ(5,1) I
      I=I+1
      IF(I(1).LT.-9999.0) STOP
C
C       NOW, COMPUTE THE TARGET ASPECT OF THE TORPEDO.
C
      R1SUM=0.0
      R2SUM=0.0
      DO 30 J=1,3
        S(J)=TO(J)-I(J)
        R2SUM=R2SUM+S(J)**2
   30 CONTINUE
      R1=SQRT(R1SUM)
      R2=SQRT(R2SUM)
      CTHETA=0.0
      DO 40 J=1,3
   40   CTHETA=CTHETA+S(J)*DVECTO(J)
      THETA=ARCOS(CTHETA/(R1*R2))
      THETA=THETA*CONV
      N(1)=DVECTO(2)*S(3)-DVECTO(3)*S(2)
      N(2)=DVECTO(1)*S(3)-DVECTO(3)*S(1)
      N(3)=DVECTO(1)*S(2)-DVECTO(2)*S(1)
      DO 50 J=1,3
   50   P2(J)=T(J)+N(J)
C
C       ROTATE THE TORPEDO ABOUT THE ARBITRARY AXIS SPECIFIED BY
C           THE TWO POINTS P2() AND S().
C
      CALL ROTATE(IR,IAXIS,I,P2,THETA)
C
C       UPDATE THE OLD DIRECTION VECTOR.
C
      R1SUM=R2SUM
      DO 60 J=1,3
   60   DVECTO(J)=S(J)
      GO TO 200
C
C       DISPLAY THE TRACK.
```

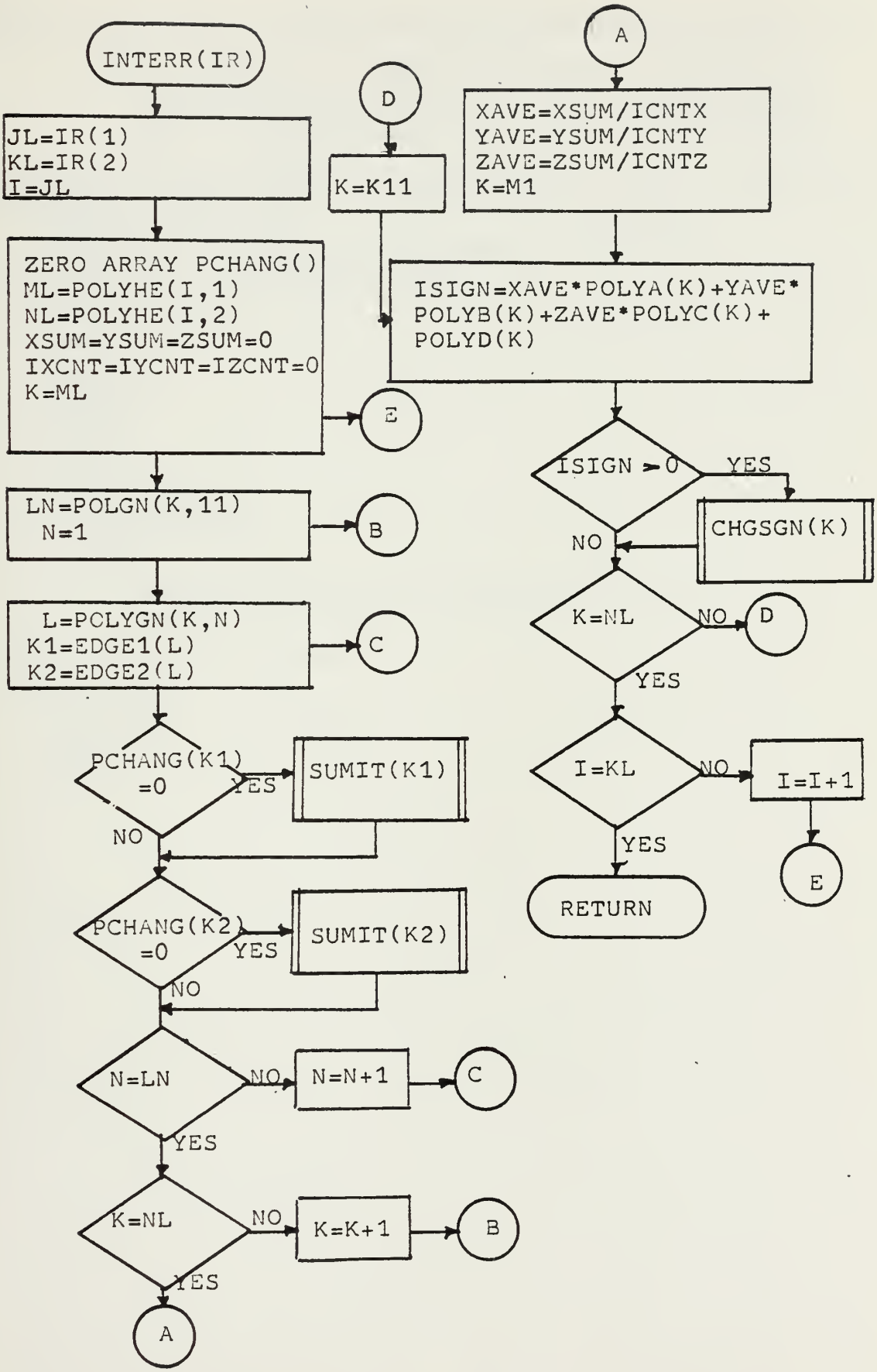
```
C
  250 IZ=I
      IP=49
      DO 70 J=1,NM
        P1(1)=TRACKX(IZ)
        P1(2)=TRACKY(IZ)
        P1(3)=TRACKZ(IZ)
        IZ=IZ-1
        IF(IZ.LT.1)IZ=5
        CALL PUTSIN(IP,J,P1)
   70 CONTINUE
      IR(1)=IPH
      IR(2)=IPH
        CALL DISPLY(IR)
      GO TO 230
C
C       THE TORPEDO HAS PENETRATED THE TEST AREA CHANGE ITS CLOR
C           AND THE TRACK'S COLOR TO RED.
C
  220 IFLAG=1
      CALL CSHAD(IR,IRED)
      GO TO 230
C
C       TORPEDO IS BACK IN THE TEST AREA SO CHANGE ITS COLOR TO BLACK.
C
  240 IFLAG=0
      CALL CSHAD(IP,ISHAD)
      GO TO 230
    1 FORMAT(3G10.3)
      END
```


MAIN
ROTATE TORPEDO TO PROVIDE INITIAL BROADSIDE ASPECT
DISPLAY TORPEDO TEST AREA'S BACK SURFACES
TRANSLATE TORPEDO TO IT'S INITIAL POSITION
IS TORPEDO BLACK
NO
YES
TORPEDO PENETRATED TEST AREA
NO
YES
CHANGE TORPEDO & TRACK TO RED
CHANGE TORPEDO'S COLOR TO BLACK
DISPLAY TORPEDO AND TRACK
READ NEW TORPEDO POSITION
T(1)< -999
STOP
COMPUTE: NEW DIRECTION VECTOR, ANGLE OF ROTATION, AXIS OF ROTATION. UPDATE TRACK
ROTATE TORPEDO

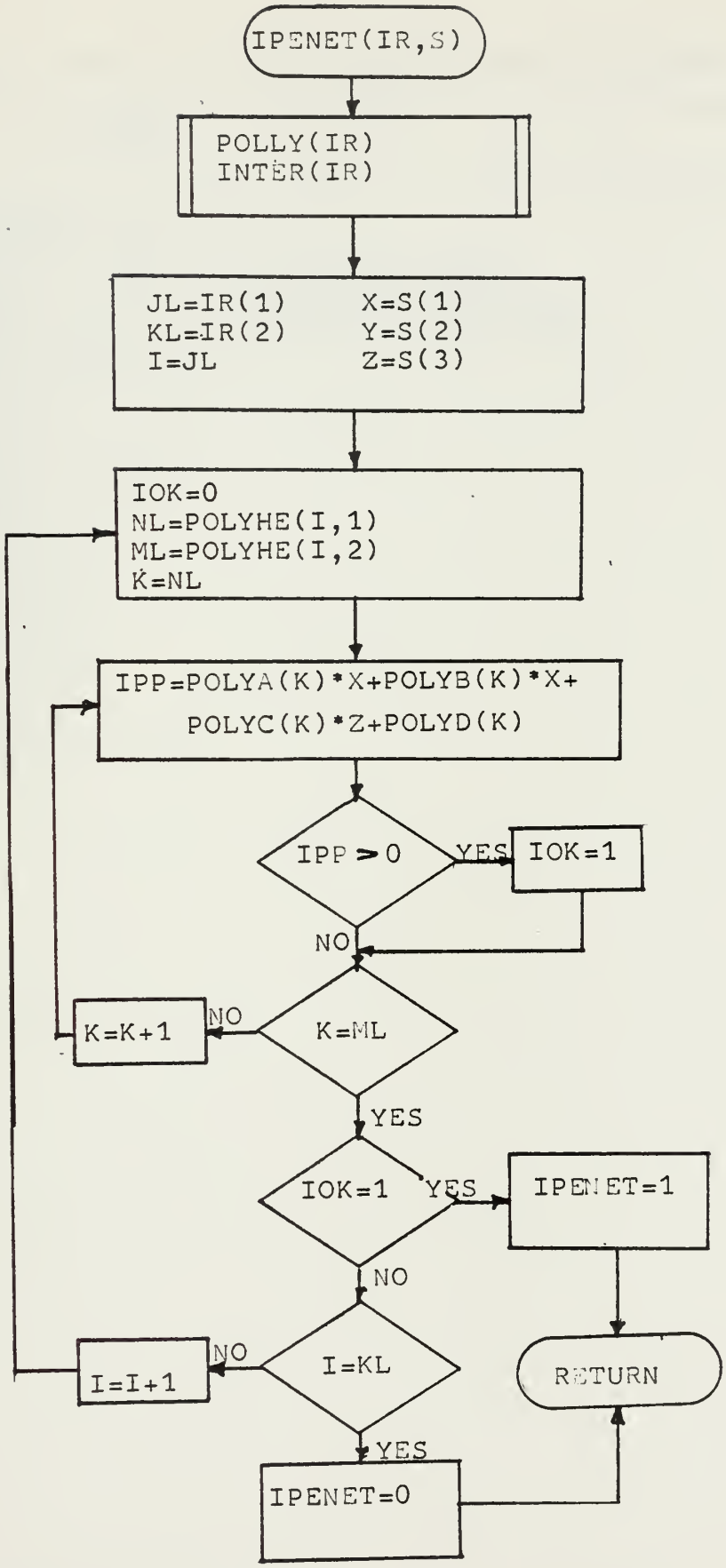
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       CHGSGN: CHANGES THE SIGN OF THE COEFFICIENTS FOR THE
C           POLYGON K.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE CHGSGN(K)
      COMMON /CA/ POLYA(60),POLYB(60),POLYC(60),POLYD(60)
      POLYA(K)=-POLYA(K)
      POLYB(K)=-POLYB(K)
      POLYC(K)=-POLYC(K)
      POLYD(K)=-POLYD(K)
      RETURN
      END
```


CSHAD(IR,NEW)
JL=IR(1)
JL=31
J=1
K=POLGN
KL=IR(2)
J=POLYHE(JL,1)
K=POLYHE(KL,2)
D O
I=J,K
SHAD(I)= NEW
RETURN

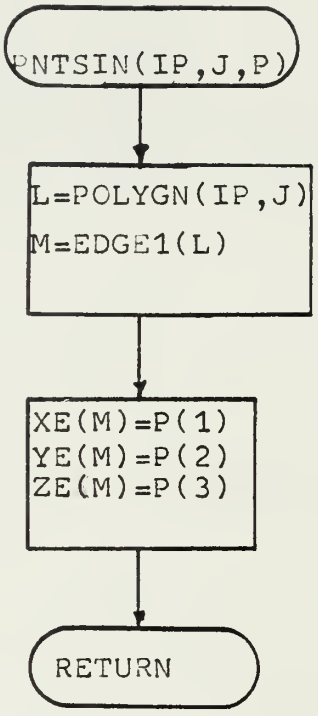
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       INTERR: COMPUTES THE POINT WHICH IS THE GEOMETRIC CENTER
C           OF A SET OF CONVEX POLYHEDRA.  THEN, IT CHANGES THE
C           SIGN OF THE COEFFICIENTS FOR ALL POLYGONS OF EACH
C           POLYHEDRON SO THAT THE DOT PRODUCT OF AN INTERIOR
C           POINT WITH THE VECTOR NORMAL WILL BE LESS THAN ZERO.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE INTERR(IR)
      DIMENSION IR(2)
      COMMON /CA/ POLYA(60),POLYB(60),POLYC(60),POLYD(60)
      COMMON /CB/ XSUM,YSUM,ZSUM,ICNT
      COMMON /AA/ POLYHE(10,2),POLYHN
      COMMON /AB/ POLYGN(60,11),POLGN,SHAD(60)
      COMMON /AC/ EDGE1(60),EDGE2(60),EDGEN
      COMMON /FF/ PCHANG(200)
      INTEGER POLYHE,POLYHN,POLYGN,POLGN,SHAD,EDGE1,EDGE2,EDGEN,
     &PCHANG
      DO 30 I=1,200
   30 PCHANG(I)=0
      JL=IR(1)
      KL=IR(2)
      DO 1200 I=JL,KL
        ML=POLYHE(I,1)
        NL=POLYHE(I,2)
        XSUM=0.0
        YSUM=0.0
        ZSUM=0.0
        ICNT=0
        DO 1210 K=NL,ML
            LN=POLYGN(K,11)
            DO 1220 N=1,LN
                L=POLYGN(K,N)
                K1=EDGE1(L)
                K2=EDGE2(L)
                IF(PCHANG(K1).EQ.0) CALL SUMMIT(K1)
                IF(PCHANG(K2).EQ.0) CALL SUMMIT(K2)
 1220       CONTINUE
 1210   CONTINUE
        C=ICNT
        XAVE=XSUM/C
        YAVE=YSUM/C
        ZAVE=ZSUM/C
        DO 1230 K=NL,ML
            ISIGN=XAVE*POLYA(K)+YAVE*POLYB(K)+ZAVE*POLYC(K)+POLYD(K)
            IF(ISIGN.GT.0) CALL CHGSGN(K)
 1230   CONTINUE
 1200 CONTINUE
      RETURN
      END
```


INTERR(IR)
JL=IR(1)
KL=IR(2)
I=JL
ZERO ARRAY PCHANG()
ML=POLYHE(I,1)
NL=POLYHE(I,2)
XSUM=YSUM=ZSUM=0
IXCNT=IYCNT=IZCNT=0
K=ML
E
LN=POLGN(K,11)
N=1
B
L=POLYGN(K,N)
K1=EDGE1(L)
K2=EDGE2(L)
C
PCHANG(K1) =0
YES
SUMIT(K1)
NO
PCHANG(K2) =0
YES
SUMIT(K2)
NO
N=LN
NO
N=N+1
C
YES
K=NL
NO
K=K+1
B
YES
A
D
K=K11
A
XAVE=XSUM/ICNTX
YAVE=YSUM/ICNTY
ZAVE=ZSUM/ICNTZ
K=M1
ISIGN=XAVE*POLYA(K)+YAVE*
POLYB(K)+ZAVE*POLYC(K)+
POLYD(K)
ISIGN > 0
YES
CHGSGN(K)
NO
K=NL
NO
D
YES
I=KL
NO
I=I+1
E
YES
RETURN

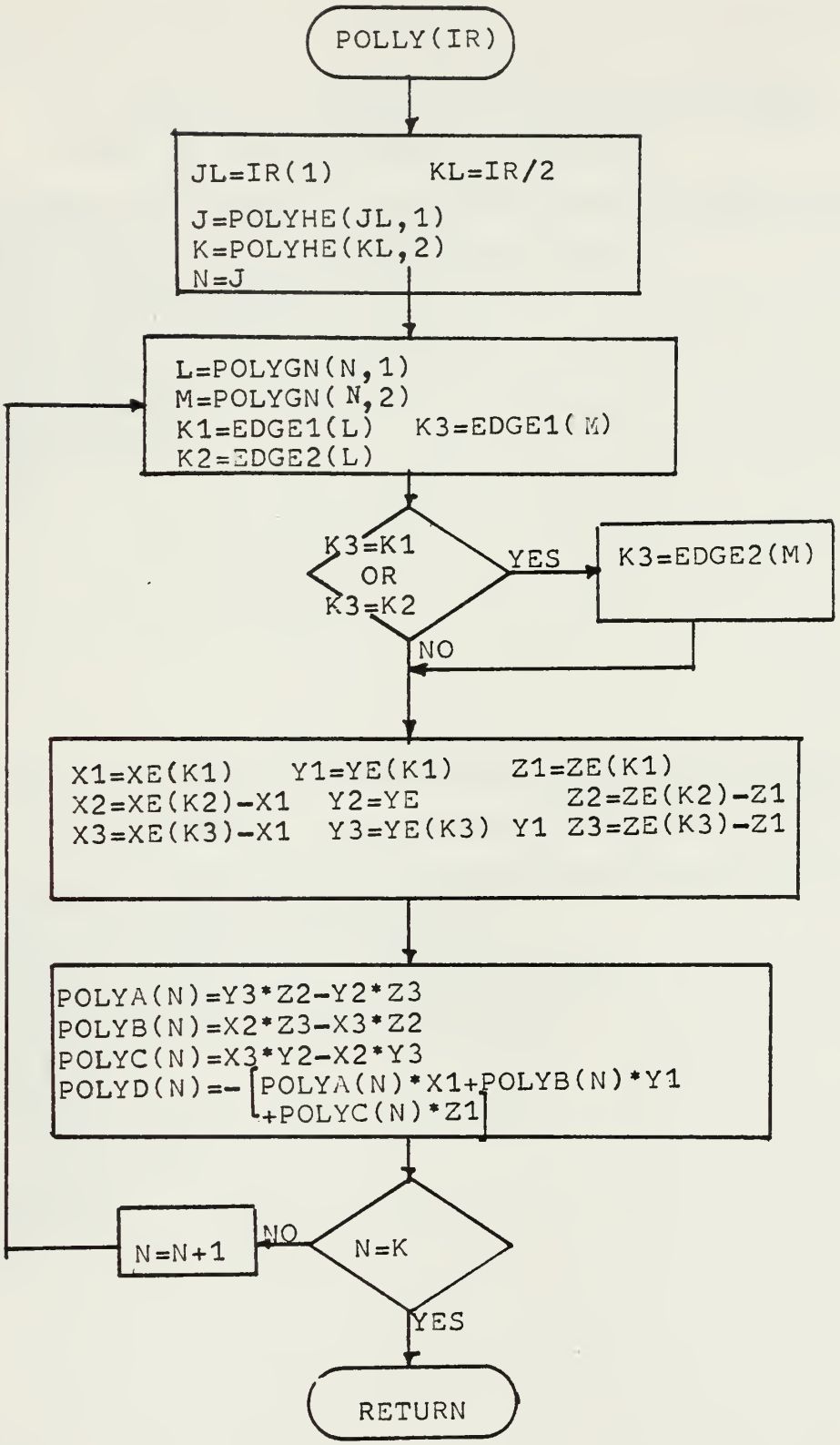
```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       IPENET: DETERMINES IF A POINT IS OUTSIDE OF A GIVEN SET OF
C           POLYHEDRA.  IF IT IS OUTSIDE OF THE POLYHEDRONS THIS
C           FUNCTION RETURNS A VALUE OF 1.  THIS FUNCTION MUST BE
C           USED IN CONJUNCTION WITH SUBROUTINES POLLY AND INTERR.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      FUNCTION IPENET(IR,S)
      DIMENSION S(3),IR(2)
      COMMON /CA/ POLYA(60),POLYB(60),POLYC(60),POLYD(60)
      COMMON /AA/ POLYHE(10,2),POLYHN
      INTEGER POLYHE,POLYHN
      CALL POLLY(IR)
      CALL INTERR(IR)
      JL=IR(1)
      KL=IR(2)
      INSIDE=0
      X=S(1)
      Y=S(2)
      Z=S(3)
      DO 1800 I=JL,KL
        NL=POLYHE(I,1)
        ML=POLYHE(I,2)
        IOK=0
        DO 1810 K=NL,ML
            IPP=POLYA(K)*X+POLYB(K)*Y+POLYC(K)*Z+POLYD(K)
            IF(IPP.GT.0) IOK=1
 1810   CONTINUE
        IF(IOK.EQ.0) INSIDE=1
 1800 CONTINUE
      IPENET=0
      IF(INSIDE.EQ.0) IPENET=1
      RETURN
      END
```


IPENET(IR,S)
POLLY(IR)
INTER(IR)
JL=IR(1)
KL=IR(2)
I=JL
X=S(1)
Y=S(2)
Z=S(3)
IOK=0
NL=POLYHE(I,1)
ML=POLYHE(I,2)
K=NL
IPP=POLYA(K)*X+POLYB(K)*X+
POLYC(K)*Z+POLYD(K)
IPP > 0
YES
IOK=1
NO
K=K+1
NO
K=ML
YES
IOK=1
YES
IPENET=1
NO
I=I+1
NO
I=KL
YES
IPENET=0
RETURN

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       PNTSIN: ALOOWS THE USER TO REPLACE ANY VEXTEX OR POINT OF
C           A POLYGON.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE PUTSIN(IP,J,P)
      DIMENSION P(3)
      COMMON /AB/ POLYGN(60,11),POLGN,SHAD(60)
      COMMON /AC/ EDGE1(100),EDGE2(100),EDGEN
      COMMON /AAA/ XE(120),YE(120),ZE(120),POINTN
      INTEGER POLYGN,POLGN,EDGE1,EDGE2,EDGEN,POINTN
      IF(J.EQ.11)GO TO 211
      NUM=POLYGN(IP,11)
      NTWO=J-1
      IF(J.EQ.1)NTWO=NUM
      L=POLYGN(IP,J)
      L2=POLYGN(IP,NTWO)
      M=EDGE1(L)
      M2=EDGE2(L2)
      XS(M)=P(1)
      XS(M2)=P(1)
      YS(M)=P(2)
      YS(M2)=P(2)
      ZS(M)=P(3)
      ZS(M2)=P(3)
      RETURN
  211 POLYGN(IP,J)=P(1)
      RETURN
      END
```

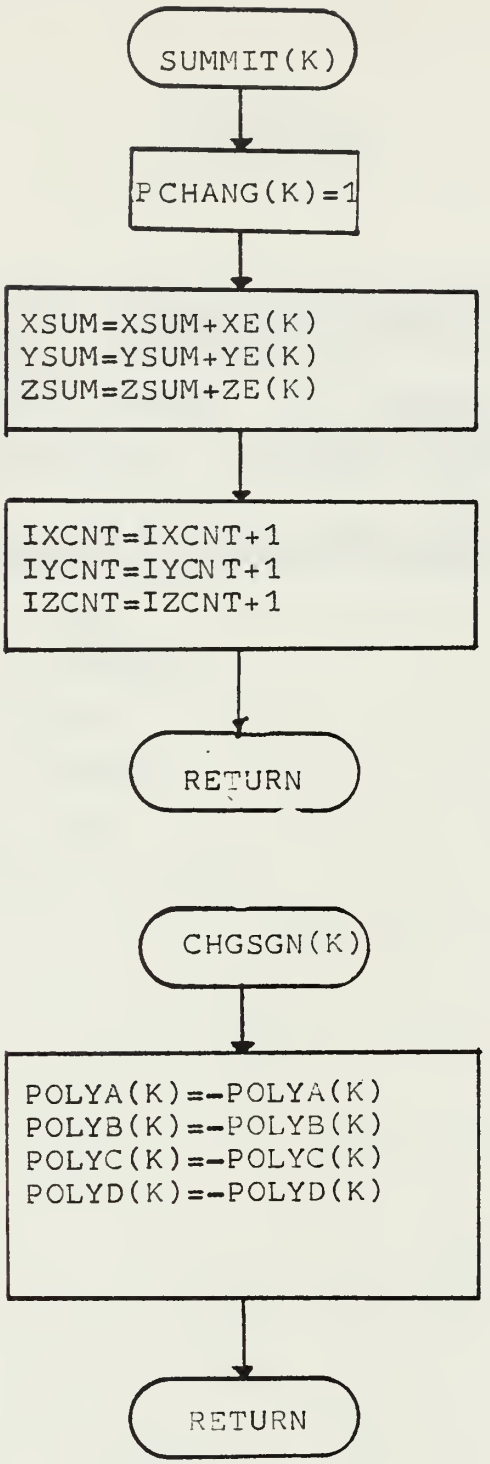

PNTSIN(IP,J,P)
L=POLYGN(IP,J)
M=EDGE1(L)
XE(M)=P(1)
YE(M)=P(2)
ZE(M)=P(3)
RETURN

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       POLLY: COMPUTES THE POLYGONAL PLANE COEFFICIENTS FOR A SET
C           OF POLYHEDRA AS DETERMINED BY THE CALLING PARAMETER IR.
C           IR(1) = THE INDEX OF THE FIRST POLYHEDRON
C           IR(2) = THE INDEX OF THE LAST  POLYHEDRON
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE POLLY(IR)
      DIMENSION IR(2)
      COMMON /CA/ POLYA(60),POLYB(60),POLYC(60),POLYD(60)
      COMMON /AA/ POLYHE(10,2),POLYHN
      COMMON /AB/ POLYGN(60,11),POLGN
      COMMON /AC/ EDGE1(100),EDGE2(100),EDGEN
      COMMON /AAA/ XE(120),YE(120),ZE(120),POINTN
      INTEGER POLYHE,POLYHN,POLYGN,POLGN,EDGE1,EDGE2,EDGEN,POINTN
      JL=IR(1)
      KL=IR(2)
      J=POLYHE(JL,1)
      K=POLYHE(KL,2)
      DO 1910 N=J,K
        L=POLYGN(N,1)
        M=POLYGN(N,2)
        K1=EDGE1(L)
        K2=EDGE2(L)
        K3=EDGE1(M)
        IF(K3.EQ.K1.OR.K3.EQ.K2) K3=EDGE2(M)
        X1=XE(K1)
        Y1=YE(K1)
        Z1=ZE(K1)
        X2=XE(K2)-X1
        Y2=YE(K2)-Y1
        Z2=ZE(K2)-Z1
        X3=XE(K3)-X1
        Y3=YE(K3)-Y1
        Z3=ZE(K3)-Z1
        POLYA(N)=Y3*Z2-Y2*Z3
        POLYB(N)=X2*Z3-X3*Z2
        POLYC(N)=X3*Y2-X2*Y3
        POLYD(N)=-(POLYA(N)*X1+POLYB(N)*Y1+POLYC(N)*Z1)
 1910 CONTINUE
      RETURN
      END
```


POLLY(IR)
JL=IR(1) KL=IR/2
J=POLYHE(JL,1)
K=POLYHE(KL,2)
N=J
L=POLYGN(N,1)
M=POLYGN(N,2)
K1=EDGE1(L) K3=EDGE1(M)
K2=EDGE2(L)
K3=K1 OR K3=K2
YES
K3=EDGE2(M)
NO
X1=XE(K1) Y1=YE(K1) Z1=ZE(K1)
X2=XE(K2)-X1 Y2=YE Z2=ZE(K2)-Z1
X3=XE(K3)-X1 Y3=YE(K3) Y1 Z3=ZE(K3)-Z1
POLYA(N)=Y3*Z2-Y2*Z3
POLYB(N)=X2*Z3-X3*Z2
POLYC(N)=X3*Y2-X2*Y3
POLYD(N)=-[POLYA(N)*X1+POLYB(N)*Y1
+POLYC(N)*Z1]
N=N+1
NO
N=K
YES
RETURN

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       SUMMIT: IS USED BY THE SUBROUTINE INTERR TO SUM THE X, Y,
C           AND Z VALUES FOR ALL OF THE VERTICES FOR THE CURRENT
C           POLYHEDRON.  ADDITIONALLY, A RUNNING COUNT OF THE
C           NUMBER OF VERTICES SUMMED IS MAINTAINED.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE SUMMIT(K)
      COMMON /AAA/ XE(120),YE(120),ZE(120),POINTN
      COMMON /CB/ XSUM,YSUM,ZSUM,ICNT
      COMMON /FF/ PCHANG(200)
      INTEGER POINTN,PCHANG
      PCHANG(K)=1
      XSUM=XSUM+XE(K)
      YSUM=YSUM+YE(K)
      ZSUM=ZSUM+ZE(K)
      ICNT=ICNT+1
      RETURN
      END
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       CSHAD: CHANGES THE SHADE OR COLOR OF ANY SET OF POLYHEDRONS.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUBROUTINE CSHAD(IR,NEW)
      COMMON /AA/ POLYHE(10,2),POLYHN
      COMMON /AB/ POLYGN(60,11),POLGN,SHAD(60)
      DIMENSION IR(2)
      INTEGER POLYHE,POLYHN,POLYGN,POLGN,SHAD
      JL=IR(1)
      KL=IR(2)
      J=POLYHE(JL,1)
      K=POLYHE(KL,2)
      DO 720 I=J,K
        SHAD(I)=NEW
  720 CONTINUE
      RETURN
      END
```


SUMMIT(K)
PCHANG(K)=1
XSUM=XSUM+XE(K)
YSUM=YSUM+YE(K)
ZSUM=ZSUM+ZE(K)
IXCNT=IXCNT+1
IYCNT=IYCNT+1
IZCNT=IZCNT+1
RETURN
CHGSGN(K)
POLYA(K)=-POLYA(K)
POLYB(K)=-POLYB(K)
POLYC(K)=-POLYC(K)
POLYD(K)=-POLYD(K)
RETURN

FCCTNOTES

[1] Phong, Bui Tuong, "Illumination for Computer Generated Pictures," Communications of the ACM, v. 16, no. 6, p.311, June 1975.

[2] Schmacker, Robert A., Sproull, Robert F., and Sutherland, Ivan E., "A Charaterization of Ten Hidden-Surface Algorithms," Computing Surveys, v. 6, no. 1, p.9, March 1974.

[3] Newman, William A. and Sprcull, Robert F., Principles of Interactive Computer Graphics, McGraw-Hill, p. 258-262, 1973.

[4] Ibid., p. 247.

[5] Ibid., p. 256-259.

[6] Ibid., p.249.

[7] Ibid., p.316-317.

[8] Ibid., p. 545.

[9] Schumacker, p. 13-14.

LIST OF REFERENCES

1. Newman, William M. and Sproul, Robert F., Principles of Interactive Computer Graphics, McGraw-Hill, 1973.

2. Appel, A. and Will, P. M., " Determing the Three-Dimensional Convex Hull of a Polyhedron", IBM Journal of Research and Developement, p. 590-601, November 1976.

3. Blin, James F. and Newell, Martin E., " Texture and Reflection in Computer Generated Images", Communications of the ACM, v. 19, no. 10, p. 542-547, October 1976.

4. Braid, I. C., " The Synthesis of Solids Bounded by Many Faces", Communications of the ACM, v. 18, no. 4, p. 209-216, April 1975.

5. Clark, James H., " Hierarchical Geometric Models for Visible Surface Algorithms", Communications of the ACM, v. 19, no. 10, p. 547-554, October 1976.

6. Crow, Franklin C., " The Aliasing Problem in Computer-Generated Shaded Images", Communications of the ACM, v. 20, no. 11, p. 799-805, November 1977.

7. Graham, N. Y., " Perspective Drawing of Surfaces With Hidden Line Elimination", The Bell System Technical Journal, v. 51, no. 4, p. 843-861, April 1972.

8. Levin, Joshua, " A Parametric Algorithm for Drawing Pictures of Solid Objects Composed of Quadratic Surfaces", Communications of the ACM, v. 19, no. 10, p. 555-563, October 1976.

9. Matsushita, Yutaka, " Hidden Line Elimination for a Rotating Object", Communications of the ACM, v. 15, no. 4, p. 245-252, April 1972.

10. McGrath, F. J., " A Method for Eliminating Hidden Lines With Polyhedra", Simulation, p. 37-41, January 1971.

11. Phong, Bui Toung, " Illumination for Computer Generated Pictures", Communications of the ACM, v. 18, no. 6, p. 311-317, June 1975.

12. Schumacker, Robert A., Sproull, Robert A., and Sutherland, Ivan E., " A Characterization of Ten Hidden-Surface Algorithms", Computing Surveys, v. 6, no. 1, p. 1-58, March 1974.

13. Teschler, Leland, " New Realism for Computer Graphics", Machine Design, p. 93-99, February 23, 1978.

14. Williamson, Hugh, " Hidden-Line Plotting Program", Communications of the ACM, v. 15, no. 2, p. 100-103, February 1972.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	Professor D. E. Kirk, Code 62Ki Chairman, Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
4.	Professor M. L. Cotton, Code 62Cc Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	3
5.	Professor R. Panholzer, Code 62 Pz Department of Electrical Engineering Naval Postgraduate School Monterey, California 93940	1
6.	Lt. Homer J. Rood, USN Rt. 8 Box 631 Pensacola, Florida 32506	1

5 AUG 80
S12238

Thesis
R698 Rood
c.1
186384
A portable three-dimensional computer graphics software package.
5 AUG 80
S12238

Thesis
R698 Rood
c.1
186384
A portable three-dimensional computer graphics software package.

thesR698
A portable three-dimensional computer gr
3 2768 001 00301 5
DUDLEY KNOX LIBRARY